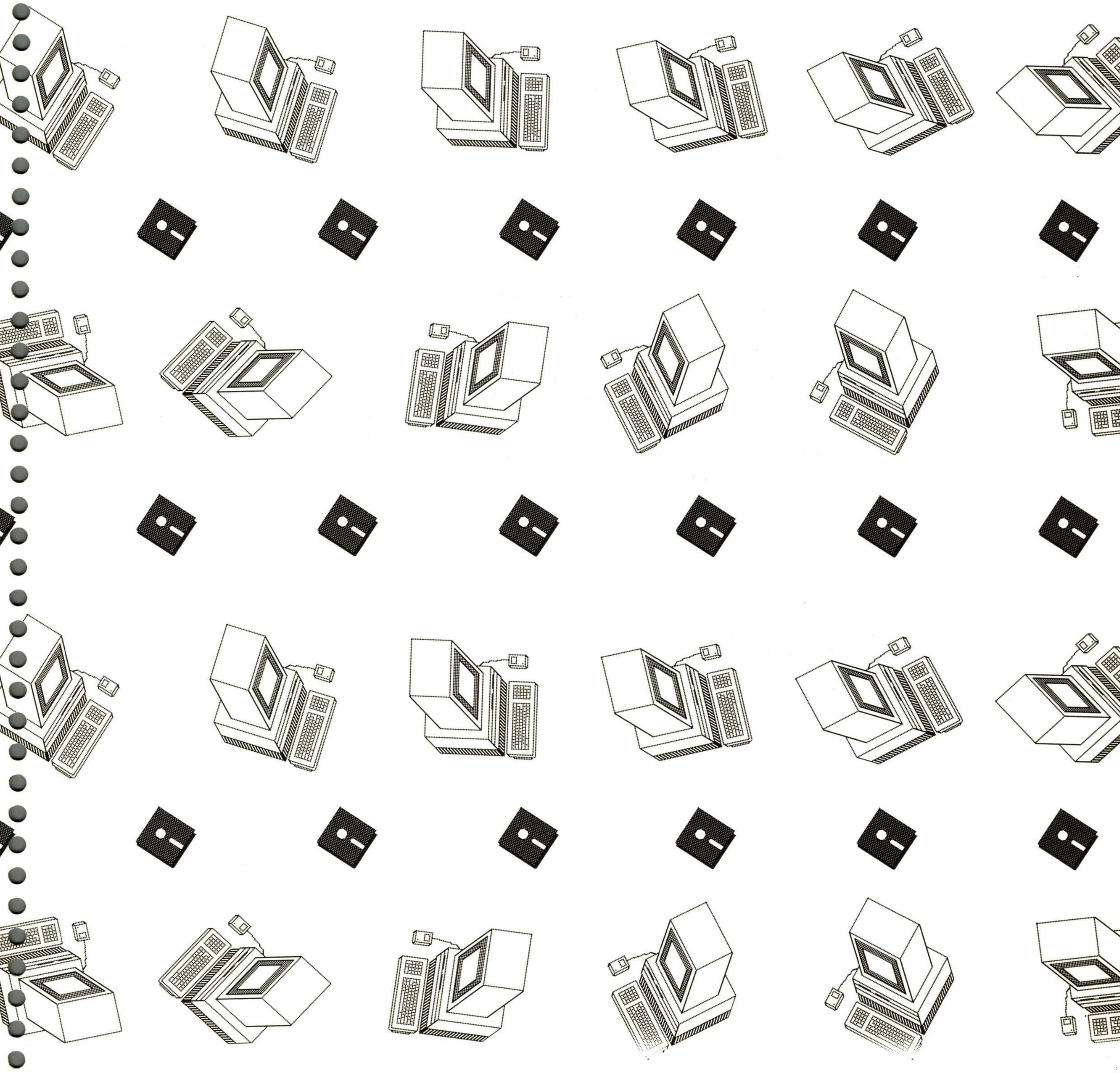
Materials per a l'assignatura d'Informàtica

Llibre tercer

Materials per a l'assignatura d'Informàtica



Llibre tercer

Aquests materials van adreçats a donar suport al desenvolupament de les assignatures següents:

- Informàtica de Segon Cicle d'Ensenyament Secundari.
- Crèdits variables de Primer Cicle d'Ensenyament Secundari.
- EATP de segon i tercer de BUP.
- Informàtica Bàsica de FP.

Autor: Joan Berga i Reixach.

- La revisió i coordinació del conjunt de materials ha estat efectuada per Jordi Orgué i Bella.

© dels autors i del PIE.

La realització ha estat a càrrec de l'Aula d'Edició Electrònica i Infografia de l'Institut Politècnic "Esteve Terradas" de Cornellà del Llobregat. Per a la seva confecció s'han utilitzat les aplicacions Framework, PageMaker, Illustrator'88 i Freehand, sobre les màquines Bull Micral i Apple Macintosh. S'han tret proves en una Laserwriter II NTX, i els fotòlits s'han obtingut en una filmadora Linotronic amb RIP de Postscript sobre materials Agfa.
Cal agraïr molt especialment l'especial deferència que ha tingut la casa HARTMANN SA, de Cornellà, per l'obtenció dels fotòlits.

Imprès a: Copisteria Dit i Fet - 788 49 90 - 08221 TERRASSA
Dipòsit legal: B-28751-90

MANUAL DE LOGO (7)

METODOLOGIA DE LA PROGRAMACIÓ (33)

PRÀCTIQUES DE LOGO (61)

Manual de Logo

1. Introducció al Logo

El llenguatge de programació Logo va néixer a finals de la dècada dels 60. Fou creat per l'equip del matemàtic Seymour Papert al Laboratori d'Intel.ligència Artificial del M.I.T. (Institut de Tecnologia de Massachusetts).

És un llenguatge de programació del tipus interpretat que posseeix les següents característiques:

- **interactiu**: les instruccions que donem s'executen immediatament i podem comprovar la seva realització.
- **procedimental**: permet treballar de forma modular descomponent els problemes complexos en blocs més senzills.
- **recursiu**: permet definir procediments que es criden a si mateixos.
- **extensible**: els procediments que definim s'incorporen al llenguatge i es poden utilitzar en la resolució d'altres problemes.

2. Conceptes generals

El Logo és un llenguatge que consta d'un nombre reduït d'instruccions, anomenades **primitives**, a partir de les quals:

- podem fer que l'ordinador realitzi coses diverses combinant aquestes instruccions.
- podem crear noves instruccions o procediments per tal que l'ordinador sàpiga fer altres coses.

Les instruccions poden ser de dos tipus: ordres i operacions.

Una **ordre** diu a l'intèrpret què ha de fer.

Una **operació** calcula un resultat.

Quan fem servir una operació, hem d'indicar què volem que faci amb el resultat.

Exemple: escriu suma 4 9

3. Posta en marxa

Els passos que cal fer per treballar amb el Logo són:

a) Carregar el sistema operatiu.

b) Carregar el llenguatge. S'ha d'escriure **logo** i prémer la tecla return.

A la pregunta "Quin espai de memòria vols reservar?" hem de respondre, simplement, prement return. Després d'uns segons, el llenguatge queda carregat.

D'altra banda, per acabar de treballar amb Logo i tornar al Sistema Operatiu s'ha d'utilitzar la primitiva **.adéu**.

4. Tipus de pantalla

El Logo distingeix tres tipus de pantalla: gràfica, de text i mixta. La pantalla gràfica permet treballar amb els gràfics de tortuga. La pantalla de text és la més idònia per treballar quan no intervenen gràfics. La pantalla mixta és com la gràfica però té 5 línies de text a la part inferior. Per passar d'una a l'altra es pot fer amb les primitives següents:

gràfic	Pantalla completa per a gràfics.
mixt	Pantalla gràfica amb 5 línies de text.
text	Pantalla per a text.

O bé, si treballem en mode directe, prement les tecles F4 (gràfica), F2 (mixta) i F1 (de text).

5. Definició i execució d'un procediment

El llenguatge Logo està pensat per treballar de forma modular, és a dir, descomponent la tasca general a realitzar en parts més petites que anomenem mòduls o "procediments". Així, doncs, la base per a l'elaboració de programes en Logo és el procediment.

Un **procediment** és un conjunt d'instruccions que donem a l'ordinador perquè realitzi una tasca determinada.

Els procediments ja definits poden servir per a la construcció de nous procediments. Això facilita l'escriptura de programes estructurats i clars, en els quals cada part té una missió concreta.

Per definir un procediment hem d'usar les primitives **procediment** i **fi**. La definició d'un procediment té la forma següent:

```
procediment nom.del.procediment
línia 1
línia 2
 .
 .
fi
```

Per executar un procediment només cal escriure el seu nom.

6. Editor de procediments

El llenguatge Logo posseeix un Editor que serveix per crear i corregir procediments. La correcció només es pot fer a l'editor. Per entrar a l'editor usarem la primitiva **edita**.

La sintaxi és: **edita** procediment/llista

Exemple: edita "mitjana

Per sortir de l'editor hem de prémer la tecla F1. En aquest cas els procediments queden definits. Si sortim prement F10, els procediments no queden definits.

7. Variables

Per poder guardar les dades que tracta un programa, necessitem usar variables. En Logo, una **variable** és una "caixa" amb un nom i un contingut (l'objecte que guarda).

El **nom** d'una variable pot ser qualsevol paraula. Les paraules de Logo han de portar cometes al davant, però no al darrera. El **contingut** d'una variable s'indica posant dos punts davant del nom de la variable.

Exemple: "edat és el nom de la variable
:edat indica el seu contingut.

Per assignar un valor a una variable ho podem fer amb la primitiva **posa.a**

Exemples:

```
posa.a "edat 18
posa.a "poble "Olot
posa.a "import.total :import1 + :import2
```

Els objectes que tracta el Logo poden ser:

- una **paraula**

  Exemples:
  "arbre
  1989

- una **llista**

  Exemples:
  [] (llista buida)
  [hola adéu Pep]
  [5 1990 -24]
  [cotxe 20 avió [blau azul]]

8. Procediments amb paràmetres

Notem que en el conjunt de primitives que té el Logo, algunes d'elles necessiten alguna dada per funcionar correctament. De la mateixa manera, els procediments que nosaltres creem poden ser amb o sense paràmetres.

Si un procediment depèn d'un o més paràmetres (variables) cal indicar-ho en el moment de la seva definició, posant-los a continuació del nom del procediment. Exemples:

```
procediment saluda :nom
escriu frase "Hola :nom
escriu [Benvingut al món del Logo!]
fi

procediment quadrat :costat
repeteix 4 [avança :costat gira.dreta 90]
fi
```

Quan vulguem utilitzar el procediment escriurem el nom seguit d'un o més objectes, que seran els que prendran les variables.

Per exemple, si ordenem quadrat 40 ens dibuixarà un quadrat de costat igual a 40 passos.

Els paràmetres d'un procediment són variables de tipus **local** i només existeixen en el moment d'execució del procediment. Totes les altres variables són **globals**, excepte si indiquem, explícitament, el contrari amb la primitiva **fes.local**, la sintaxi de la qual és:

fes.local "variable

9. Primitives d'entrada i de sortida

* Entrada de dades: Per entrar un valor (numèric o alfanumèric) hem d'usar la primitiva **paraula.llegida**. Aquest valor s'ha de guardar en una variable mitjançant una assignació.

La sintaxi completa és:

posa.a "variable **paraula.llegida**

Exemple: posa.a "costat paraula.llegida
posa.a "nom paraula.llegida

* Sortida de dades. Per fer sortir un valor (numèric o alfanumèric) per la pantalla hem d'usar la primitiva **escriu**.

Exemples: escriu :edat
escriu "Barcelona
escriu [hola, com estàs?]

10. Gestió de l'espai de treball

Anomenem **espai de treball** a tot el que es troba a la memòria de l'ordinador quan estem treballant amb el Logo. Vegem algunes primitives que serveixen per tractar o gestionar aquest espai de treball.

títols?	Mostra el nom de tots els procediments que hi ha a l'espai de treball.
presenta proc/llis	Imprimeix el procediment o la llista de procediments.
oblida proc/llista	Esborra els procediments que indiquem.
oblida.tot	Esborra tot el contingut de l'espai de treball.

11. Gestió dels arxius de procediments

Comentarem aquí les primitives que ens permeten guardar en el disquet i recuperar del disquet el treball efectuat amb Logo.

desa arxiu llista	Guarda a l'arxiu indicat els procediments de la llista.
recupera arxiu	Carrega a la memòria l'arxiu indicat.
arxius? unitat	Dóna un llistat dels arxius del disquet.
elimina arxiu	Esborra del disquet l'arxiu indicat.

12. Expressions lògiques

Com d'altres llenguatges de programació, Logo permet tractar expressions lògiques. Aquestes, com ja sabem, poden ser simples o compostes. El Logo té els valors lògics: **ver** i **fals**.

Els **operadors de comparació** són: = , < i >

Exemples:

:preu < 1200
:nom = "Pere
:edat = edat.màxima

Els **operadors lògics** són:

no predicat	Retorna el valor contrari del predicat.
veres.totes p1 p2	Retorna ver si ambdós predicats són certs, sinó retorna fals.
vera.alguna p1 p2	Retorna ver si algun dels predicats és cert, sinó retorna fals.

Exemples:

no :comptador = 10
veres.totes (:costat<20) (:n=8)

13. Estructura condicional

L'estructura condicional té la forma:

si condició llista1 llista2

Si la condició és certa s'executa la llista1, sinó s'executa la llista2. La llista2 és optativa.

Exemples:

si :costat > 100 [escriu [número massa gros] acaba]

si :edat < 18 [escriu [menor d'edat]] [escriu [major d'edat]]

D'altra banda convé esmentar dues primitives lligades amb el control d'execució dels procediments. Són:

acaba	Acaba l'execució d'un procediment.
espera n	Fa una pausa d'uns n/30 segons.

14. Estructura iterativa o repetitiva

El Logo té dues maneres per fer una estructura repetitiva. Convé triar-ne una o altra segons el que vulguem fer.

Si d'entrada es coneix el nombre de vegades que cal repetir usarem la primitiva repeteix.

La sintaxi és: **repeteix** n [accions]

Exemple:

```
repeteix 4 [escriu frase :n 2*:n posa.a "n :n+1]
```

Una altra forma consisteix a usar la recursivitat de cua. Vol dir que el procediment es crida a si mateix a l'última línia.

Obviament, hi ha d'haver una línia de control per tal que el procediment no esdevingui interminable.

Exemple:

```
procediment cubs.menors.de.10
posa.a "num 1
cub :num
fi

procediment cub :n
si :n > 10 [acaba]
escriu frase :n :n*:n*:n
cub :n+1
fi
```

15. Funcions

Recordem que en Logo hi ha dos tipus de primitives: ordres i operacions. També podem considerar dos tipus de procediments: els que fan una tasca i els que retornen un valor. Aquests últims es coneixen amb el nom de procediments de tipus funció o, simplement, funcions. Per definir una **funció** hem d'utilitzar la primitiva:

retorna objecte	Acaba un procediment i retorna objecte com a resultat.

Exemple:

```
procediment mitjana :num1 :num
retorna (:num1 + :num2) / 2
fi
```

Ara si fem mitjana 5 9 ens donarà el valor 7. Recordem també amb el resultat d'una operació o d'una funció hem de fer alguna cosa. Pot formar part d'una expressió o bé s'ha d'escriure per pantalla.

16. Recursivitat

Ja hem dit que el Logo és un llenguatge de programació recursiu, és a dir, un procediment pot cridar-se a si mateix. Podem distingir dos tipus de recursivitat: de cua i central.

La recursivitat de cua, que ja hem esmentat abans, és quan un procediment es crida a si mateix a l'última línia. S'utilitza moltíssim en Logo com una eina per a la repetició o iteració de blocs d'instruccions.

Exemple:

```
procediment circumferència
avança 1
gira.dreta 1
circumferència
fi
```

Per recursivitat central s'entén quan la línia on el procediment es crida a si mateix no és l'última. En aquest cas s'obtenen procediments amb unes possibilitats importants i, moltes vegades, sorprenents. Compareu, si més no, aquests dos exemples que donem.

Exemples:

```
procediment compte.avall :n
si :n < 1 [acaba]
escriu :n
compte.avall :n-1
fi
```

```
procediment sorpresa :n
si :n < 1 [acaba]
sorpresa :n-1
escriu :n
fi
```

17. Primitives elementals dels gràfics de tortuga

Un dels ambients Logo que ha fet més popular aquest llenguatge és el món dels **gràfics de tortuga**. La tortuga és un punt orientat que pot ser controlat amb ordres directes o bé amb l'execució de procediments prèviament definits. El desplaçament de la tortuga per la pantalla de l'ordinador ens permet fer gràfics. Les instruccions més elementals que podem donar a la tortuga són:

avança n	Avança n passos.
recula n	Recula n passos.
gira.dreta n	Gira a la dreta n graus sexagesimals.
gira.esquerra n	Gira a l'esquerra n graus sexagesimals.
apareix	Mostra la tortuga.
desapareix	Amaga la tortuga.
no.llapis	La tortuga no dibuixa per allà on passa.
llapis	La tortuga dibuixa per allà on passa.
goma	La tortuga esborra allò que troba dibuixat.
inicia.dibuix	Per netejar la pantalla gràfica.

18. El món matemàtic

Un altre ambient del llenguatge Logo és el **món matemàtic**. Logo pot treballar amb números enters, decimals i en notació científica. Les operacions poden tenir la forma prefixada o infixada. D'altra banda, el Logo posseeix una sèrie de funcions matemàtiques ja definides en el propi llenguatge.

Els símbols i les primitives per indicar les operacions i funcions matemàtiques són:

Operacions (forma infixada):

+	Suma
-	Resta
*	Multiplicació
/	Divisió

Operacions (forma prefixada):

suma n1 n2	Retorna la suma de n1 i n2.
resta n1 n2	Retorna la diferència de n1 i n2.
producte n1 n2	Retorna el producte de n1 i n2.
divisió n1 n2	Retorna la divisió de n1 per n2.
quocient n1 n2	Retorna el quocient enter entre n1 i n2.
arrel n	Retorna l'arrel quadrada de n.

Funcions:

part.entera n	Retorna la part entera de n.
arrodonit n	Retorna el número n arrodonit.
residu n1 n2	Retorna la resta de la divisió de n1 i n2.
potència n a	Retorna n elevat a la potència a.
exp n	Retorna e elevat a la potència n.
ln n	Retorna el logaritme neperià de n.
sin n	Retorna el sinus d'un angle de n graus.
cos n	Retorna el cosinus d'un angle de n graus.
arctg n	Retorna en graus l'arctangent de n.
atzar n	Retorna un enter i positiu entre 0 i n-1.

19. Paraules i llistes

Un tercer ambient del llenguatge Logo és el **món de les paraules i de les llistes**. Precisament, en principi, el Logo era un llenguatge pensat per manipular objectes (paraules i llistes). Més endavant s'afegiren els gràfics, que l'han fet famós.

En Logo cal distingir entre paraula, frase i llista. Una paraula és una successió de caràcters entre els quals no hi ha cap espai en blanc. Una llista és una successió d'elements que poden ser paraules o llistes. Una frase és un cas particular de llista en la qual tots els elements són paraules. Les paraules s'escriuen amb unes cometes al començament. Les llistes s'escriuen entre claudàtors.

Les primitives que s'usen en el tractament de paraules i llistes són:

ascii paraula	Retorna el codi ASCII del primer caràcter de la paraula.
caràcter n	Retorna el caràcter n-èssim del codi ASCII.
n.elements obj	Retorna el número d'elements d'objecte.
element n obj	Retorna l'element que ocupa la posició n.
paraula obj1 obj2	Retorna la paraula formada per obj1 i obj2.
frase obj1 obj2	Retorna una llista formada pels elements d'obj1 i d'obj2.
llista obj1 obj2	Retorna una llista amb els elements obj1 i obj2.
és.nombre obj	Retorna ver si obj és un número.
és.paraula obj	Retorna ver si obj és una paraula.
és.llista obj	Retorna ver si obj és una llista.
són.iguals ob1 ob2	Retorna ver si ob1 i ob2 són iguals.
és.buida obj	Retorna ver si obj és la paraula o la llista buida.
pertany obj ll	Retorna ver si obj és un element de llista.
sense.primer obj	Retorna l'objecte menys el primer element.
sense.últim obj	Retorna l'objecte menys l'últim element.
anteposant obj ll	Retorna una llista resultant d'afegir l'objecte

	davant de la llista.
posposant obj ll	Retorna una llista resultant d'afegir l'objecte darrera de la llista.
primer obj	Retorna el primer element de l'objecte.
últim obj	Retorna l'últim element de l'objecte.

on ob i obj indiquen objecte i ll indica llista.

20. Comunicació amb el món exterior

En aquest apartat veurem algunes primitives més d'entrada i de sortida i d'altres que ens permeten millorar la presentació dels resultats a la pantalla de l'ordinador. Són:

escriu objecte	Escriu objecte i passa a la següent línia.
escriu.s objecte	Escriu objecte i després no salta de línia.
car.llegit	Retorna el caràcter llegit del teclat.
cars.llegits n	Retorna els n caràcters llegits del teclat.
paraula.llegida	Retorna la paraula llegida del teclat.
llista.llegida	Retorna la llista llegida del teclat.
esborra.text	Esborra la pantalla de text i posa el cursor al principi de la primera línia.
cursor	Retorna les coordenades del cursor.
mou.cursor [x y]	Situa el cursor a la posició (x,y).

21. Altres primitives dels gràfics de tortuga

En usar els gràfics de tortuga hem de tenir present que també podem treballar amb coordenades cartesianes amb l'origen al centre de la pantalla. El mode gràfic habitual té 640x200 punts, la qual cosa representa 640 passos de tortuga d'amplada i uns 440 d'alçada. Altres primitives de l'ambient gràfic són:

orientació	Retorna l'orientació de la tortuga en graus sexagesimals.
posició	Retorna la posició de la tortuga en coordenades cartesianes.
coor.x	Retorna la coordenada X de la tortuga.
coor.y	Retorna la coordenada Y de la tortuga.
orienta't n	Posa la tortuga en l'orientació n.
posa't [x y]	Situa la tortuga a la posició (x,y).
fes.x n	La tortuga va a la posició (n,coor.y).
fes.y n	La tortuga va a la posició (coor.x,n).
a.casa	Situa la tortuga a l'origen de coordenades.
esborra.dibuix	Neteja la pantalla sense moure la tortuga d'on està.
punt [x y]	Dibuixa un punt sobre la pantalla gràfica a la posició (x,y), sense moure la tortuga.

Tipus de pantalla gràfica:

tor	En sortir la tortuga per un costat de la pantalla continua pel costat oposat.
finestra	La tortuga pot moure's fora de la pantalla que ve a ser com una finestra d'observació.
barrera	La tortuga no pot sortir dels límits de la pantalla.

22. Altres primitives per al control de l'execució

Veurem ara algunes primitives que permeten definir l'estructura condicional d'una forma diferent a la que ja coneixem i alguna més que també serveix per al control de l'execució. Són:

executa llista	Executa les instruccions de la llista.
comprova pred	Comprova si el predicat és ver o fals.
si.ver llista	Executa les instruccions de la llista si l'últim comprova era ver.
si.fals llista	Executa les instruccions de la llista si l'últim comprova era fals.

23. Altres estructures de control

El Logo no posseeix totes les estructures de control vistes als apunts de Metodologia de la Programació. Per això donem els procediments que us permetran disposar d'algunes d'aquestes estructures algorísmiques compostes:

* Estructura iterativa tipus "**mentre**"

```
procediment mentre :condició :llista_d'accions
si no executa :condició [acaba]
executa :llista_d'accions
mentre :condició :llista_d'accions
fi
```

* Estructura iterativa tipus "**fins que**"

```
procediment fins.que :condició :llista_d'accions
executa :llista_d'accions
si executa :condició [acaba]
fins.que :condició :llista_d'accions
fi
```

* Estructura iterativa tipus "**indexat**"

```
procediment per :var :inici :final :inc :llista_d'accions
si :inici > :final [acaba]
posa.a :var :inici
executa :llista_d'accions
per :var :inici + :inc :final :inc :llista_d'accions
fi
```

* Estructura iterativa tipus "**indexat**" amb l'increment igual a 1

```
procediment per :var :inici :final :llista_d'accions
si :inici > :final [acaba]
posa.a :var :inici
executa :llista_d'accions
per :var :inici + 1 :final :llista_d'accions
fi
```

* Estructura condicional múltiple

Aquesta estructura es pot resoldre amb una sèrie de condicionals simples de la forma:

```
posa.a "control "fals
si :var = v1 [llista_d'accions 1 posa.a "control "ver]
si :var = v2 [llista_d'accions 2 posa.a "control "ver]
......
si :var = vn [llista_d'accions n posa.a "control "ver]
si no :control [llista_d'accions]
```

Metodologia de la Programació

1. Introducció

1.1. Problemes, algorismes i programes

En aquest tema d'Introducció a la Programació hem de veure la informàtica com una eina per col.laborar en la resolució d'un **problema** determinat. Ara bé, un cop plantejat el problema, el primer pas a realitzar és, de fet, extrainformàtic. Es tracta, en primer lloc, de buscar un mètode per a la resolució del problema.

Aquest mètode ha de ser general i descrit de forma simple, clara i no ambigua.

Quan descrivim els passos que cal fer per resoldre un problema es pot dir que hem definit un **algorisme** de resolució d'aquest problema.

Notem que de qualsevol tasca, més o menys complexa, se'n pot escriure un algorisme. Per exemple: canviar una roda rebentada d'un cotxe, fer una truita a la francesa, trobar el volum d'una habitació, ordenar una llista de 10 números, etc. Hi ha, però, algorismes que donen lloc a programes i d'altres que no, perquè el que plantegen són feines que no pot fer un ordinador.

S'entén, doncs, que partim d'un **problema** que volem resoldre amb l'ordinador; la manera com resoldre'l dóna lloc a l'**algorisme** i, finalment, les instruccions que donem a l'ordinador constitueixen el **programa**.

És a dir, per resoldre un problema mitjançant un programa d'ordinador fan falta:

1) Tenir molt clar l'enunciat del problema (el problema).

2) Trobar un mètode de resolució general (l'algorisme).

3) Escriure aquest mètode en forma d'instruccions que puguin ser interpretades per l'ordinador (programa).

1.2. Concepte de programa

Coneixem el significat de la paraula programa en la vida ordinària. Per exemple, parlem del programa d'actes de la Festa Major, de la programació de la televisió, etc. Per als informàtics, el sentit de la paraula és semblant.

En informàtica, un **programa** és el conjunt d'instruccions que donem a l'ordinador per indicar-li les coses que ha de fer.

El programa, així com les dades, s'introdueix mitjançant els perifèrics (d'entrada o d'emmagatzematge) i es guarda a la memòria principal. Més tard, la Unitat de Control s'encarrega de seguir aquestes instruccions una a

una. Val a dir que un ordinador sense cap programa carregat a la memòria no és capaç de fer res.

Per donar un exemple no informàtic, podem dir que un programa s'assembla molt a una recepta de cuina, on s'hi explica els passos que s'han de fer per obtenir un cert menjar. El funcionament d'un ordinador és similar al d'una cuina, on una persona que no sap cuinar (unitat de control) té anotades en una pissarra (memòria) unes instruccions molt detallades (programa) que li expliquen com cuinar els aliments (les dades) per tal d'obtenir diversos plats (els resultats). Per una finestra (unitat d'entrada) li donem els aliments (dades) i les instruccions (programa). Després del procés ens torna els plats cuinats (resultats) per una altra finestra (unitat de sortida).

Recordem que per aconseguir que un ordinador realitzi alguna tasca, cal entrar-li, prèviament, un programa que alguna persona haurà hagut de pensar.

1.3. Complexitat de la programació. Necessitat d'un mètode

En principi, podríem parlar de la programació com el camí (a vegades molt llarg) que condueix des d'un problema (generalment mal plantejat) a una solució (no sempre satisfactòria). Aquesta solució s'obté a través de l'execució d'un programa. Convé fer notar, doncs, que es tracta d'obtenir un resultat, no un programa.

Programar correctament no és una feina fàcil. Cal analitzar molt profundament quin és el problema, quines són les dades de què es disposa, quin són els resultats que volem obtenir i, a més, cal preveure totes les situacions estranyes que es poden produir al llarg del procés.

Hem de tenir en compte, per exemple, que l'ordinador fa sempre una feina darrera l'altra (no en pot fer dues alhora), només entén ordres senzilles i no resol mai un imprevist.

Quan vulguem escriure un programa, convé, doncs, seguir un mètode. El primer que cal fer és entendre bé el problema que volem resoldre. Després s'ha de descomposar el treball a realitzar en treballs simples, aquests en d'altres més simples successivament, fins a arribar a treballs veritablement senzills.

També és important seguir un mètode en la programació perquè la resta de persones entengui el que fem i nosaltres mateixos ho entenguem més endavant.

Hi ha molts estudis fets sobre les conseqüències catastròfiques que es deriven de la falta d'un mètode.

D'altra banda, només té sentit programar algorismes ja coneguts i comprovats. Ningú no ha de posar-se a programar un problema sense conèixer el

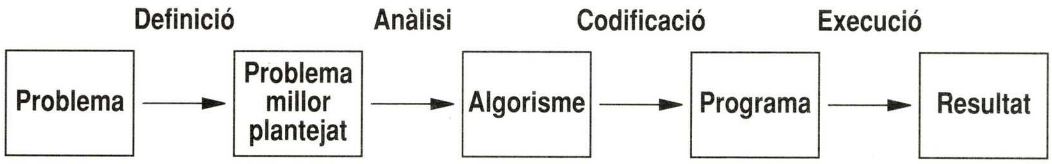
mètode de resolució que pensa fer servir (l'algorisme). Programar directament un problema no és gens aconsellable. Per programar bé, no n'hi ha prou al dominar el llenguatge de programació que fem servir (Pascal, Logo, Basic, etc.), abans, cal conèixer el mètode de resolució del problema. Només té sentit escriure un programa un cop hàgim escrit l'algorisme de resolució del problema.

En aquest sentit, hem de veure un programa com l'expressió d'un algorisme en un llenguatge entenedor per a l'ordinador.

1.4. Etapes en l'elaboració d'un programa

Per a l'obtenció d'un programa s'han d'efectuar les següents etapes:

1) Definició del problema.
2) Anàlisi.
3) Codificació.
4) Explotació.



Anem a veure què significa cada una d'elles.

* Definició del problema

Aquesta fase del desenvolupament de programes consisteix a comprendre perfectament les especificacions del problema que hem de resoldre, és a dir, tenir molt clar l'enunciat del problema i el tipus de solució que es desitja.

Podríem dir que en aquest punt estudiem el **QUE** hem de fer, i això es materialitza en la redacció d'una sèrie de notes, documents o dossiers, a partir de les que s'elabora el manual de l'usuari.

* Anàlisi

Es tracta de trobar un mètode per resoldre el problema que hem definit en l'etapa anterior. Aquest mètode ha de ser general i ha de preveure tots els casos que es puguin donar.

És evident, també, que, un cop trobat el mètode, caldrà comprovar la seva validesa mitjançant una exploració, execució "manual" d'alguns casos comprovant, de manera teòrica, que la lògica del mètode obté els resultats desitjats.

Aquí estudiem el **COM** fer-ho. La manera d'efectuar l'anàlisi d'un problema i les eines que s'usen per a aquesta tasca s'expliquen més endavant.

La persona que realitza aquesta feina rep el nom d'**analista**.

* Codificació

Es tracta d'escriure els diferents passos del mètode anterior en una sèrie d'instruccions comprensibles per a l'ordinador. Aquestes constituiran el programa que introduirem a la màquina.

Això s'aconsegueix mitjançant els llenguatges de programació que vàrem tractar en el tema "El Sistema Informàtic". L'escriptura d'un programa en un llenguatge de programació és l'últim pas a realitzar i moltes vegades el més senzill. La persona encarregada d'aquesta tasca s'anomena **programador**.

* Explotació

Consisteix a comprovar i usar el programa ja codificat en l'equip corresponent per part dels **usuaris**.

• • • • •

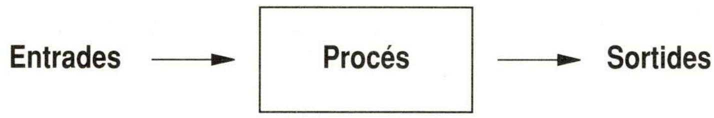
La primera part del procés, la que ens porta del problema a l'algorisme, és la més important, alhora que la més delicada. Si es cometen errors en aquesta part i no es detecten a temps, això implica, generalment, que s'haurà de començar altre cop des del principi.

Cal fer notar que aquesta primera part de la programació és pràcticament independent del llenguatge i de la màquina que es farà servir posteriorment. Un cop es té l'algorisme, aplicant regles de codificació senzilles, s'acaba el programa d'una forma quasi mecànica. Cal insistir que aprendre un llenguatge de programació no vol dir aprendre a programar.

1.5. Procés: dades i algorisme

Un sistema és un conjunt d'elements que evolucionen segons unes lleis per portar a terme alguna funció.

L'esquema d'un sistema és:



En tot sistema hi ha uns objectes (les dades) i una activitat (el procés). També hem de considerar les entrades i les sortides.

És a dir, en un sistema tenim:

- Dades: Els objectes que es manipulen.
- Procés: L'activitat interna del sistema.
- Entrades: Les dades que es lliuren per al seu tractament.
- Sortides: Les dades que s'obtenen com a resultat del procés.

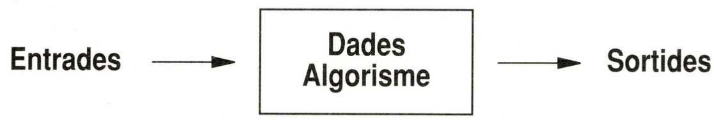
Recordem que ja havíem parlat de l'ordinador com un sistema informàtic. Concretament, dèiem que l'ordinador és un sistema per al tractament de dades.

En tot procés convé distingir les dades i l'algorisme.

Les dades són els objectes que manipula el sistema.

L'algorisme és l'especificació de l'activitat, és a dir, el conjunt de passos que cal seguir per realitzar l'activitat.

Així, doncs, l'esquema d'un sistema informàtic és:



2. Dades

2.1. Introducció

Recordem que un sistema informàtic tracta dades i que en un procés hem de considerar les **dades** i l'**algorisme**.

La importància actual dels ordinadors es deu al fet que permeten el tractament de grans quantitats d'informació. La informació és una abstracció (simplificació) de la realitat. La informació que tracta l'ordinador és una selecció de dades de la realitat, concretament aquelles que són importants per al problema que tractem. Per exemple, el fitxer del personal d'una empresa només conté les dades que interessen: nom, adreça, telèfon, número de la Seguretat Social, categoria professional, etc.

Les dades són per a la informàtica com els ingredients per a la cuina, és a dir, la base de tota realització.

Ara bé, les dades que tracta un ordinador no poden estar desordenades. Cal anomenar-les d'alguna manera, manipular-les, etc. Convé que hi hagi una certa estructura en la seva organització. És a dir, s'ha d'organitzar la memòria relativa a un problema, segons les necessitats del mateix, en una estructura que es coneix com a **estructura de dades**.

Les dades s'agrupen en classes (dites també "tipus"), els elements de les quals posseeixen, tots, les mateixes propietats.

2.2. Constants i variables

De manera intuïtiva, o per matemàtica, tenim el concepte de constant i de variable. En informàtica considerem:

Constant: és una dada que no varia al llarg de tot el procés.

Variable: és un cert lloc de la memòria, amb un nom o identificador, que serveix per guardar les dades amb les quals ha de treballar l'ordinador.

Una variable es pot comparar a un calaix al qual hem donat un nom i on podem guardar coses. També podem comparar-la a un apartat de correus o a cada una de les caixetes del clauer d'un hotel.

Cal remarcar que:

- Les dades que conté una variable poden variar al llarg del procés.
- Per identificar una variable cal donar-li un nom.
- Una variable només guarda l'última cosa que li hem posat.

En matemàtica es classifiquen les variables d'acord a certes característiques o propietats comunes. Es fa distinció entre les variables reals, complexes, etc., o entre variables que representen valors individuals o conjunts de valors. Aquesta noció de classificació també és molt important en informàtica.

2.3. Operadors

Són una sèrie de símbols utilitzats per indicar les operacions matemàtiques, les relacions o comparacions matemàtiques i lògiques i l'emmagatzematge d'un cert valor en una variable.

Hi ha quatre tipus d'operadors: matemàtics, de comparació, lògics i d'assignació.

2.3.1. Operadors matemàtics

Serveixen per indicar les operacions aritmètiques elementals.

Són:

+	Suma
-	Resta
*	Multiplicació
/	Divisió
^	Potenciació

Recordem que també es poden usar (i de vegades cal fer-ho obligatòriament) els parèntesis per escriure una expressió matemàtica.

Exemple: $\frac{A}{B + C}$ ha de ser A/(B+C)

2.3.2. Operadors de comparació

Serveixen per realitzar una comparació entre constants, variables o expressions. Són:

=	Igual
<	Més petit que
<=	Més petit o igual que
>	Més gran que
>=	Més gran o igual que
<>	No igual a

S'usen en les estructures de control que veurem més endavant.

Exemples: 6 < 8   A <= B   C <> 5

2.3.3. Operadors lògics

Serveixen per indicar una condició composta, formada a partir d'una combinació de condicions simples. Les més importants són:

i **Conjunció**: la condició composta s'acompleix si s'acompleixen cada una de les condicions elementals.

Exemples: En Joan canta i la Nuri balla
A>200 i ofici="paleta"

o **Disjunció**: la condició composta s'acompleix si almenys s'acompleix alguna de les condicions elementals.

Exemples: plou o fa vent
preu<2000 o Q=10

no **Negació**: la condició composta s'acompleix sempre que no s'acompleixi la condició elemental.

Exemples: no fa sol
no (B<15)

Les taules de la veritat són:

P	Q	P i Q
V	V	V
V	F	F
F	V	F
F	F	F

P	Q	P o Q
V	V	V
V	F	V
F	V	V
F	F	F

P	no P
V	F
F	V

2.3.4. Operador d'assignació

Serveix per indicar que una constant, el contingut d'una variable o el resultat d'una expressió es guarda en una altra variable.

S'indica amb el símbol <-   Exemples: A <- 5 ; nom <- "Pere"

2.4. Tipus de dades

2.4.1. Concepte de tipus

Les dades poden ser de diverses classes o tipus (per exemple: numèriques, de text, etc.).

En informàtica interessa diferenciar els tipus de dades ja que els compiladors precisen, per a optimitzar l'execució dels programes, conèixer els tipus de dades amb que treballaran.

Un tipus de dades defineix una estructura, els valors que pot agafar i les operacions que s'hi poden realitzar. Per exemple, al parlar del tipus caràcter sabem la memòria que ocupa, els valors que pot agafar i les operacions que s'hi poden efectuar.

Cada tipus de dades porta associat una sèrie d'operacions pròpies, a més dels operadors bàsics comuns a qualsevol tipus, com la comparació (d'igualtat o d'ordre per als tipus ordenats) i l'assignació.

Una constant és un valor particular del conjunt dels valors del seu tipus.

En una variable podem considerar:

identificador: el seu nom.
tipus: al qual pertany.
valor: la dada que guarda en un moment determinat.

En una constant tindrem:

identificador: el seu nom.
tipus: al qual pertany.
valor: la dada que guarda de forma permanent.

2.4.2. Tipus de dades

Les dades les podem classificar des de diferents punts de vista, depenent de l'aprofundiment en què vulguem tractar el tema. Podem veure:

A) Una primera classificació senzilla pot ser:

- **Numèriques**: representen un valor numèric.

  Exemples: 25 , -642 , 53.18 , 2.35E+08

Cal tenir present que si la dada numèrica és molt grossa o molt petita l'ordinador l'expressarà en notació científica.

- **Alfanumèriques** o **literals**: és un conjunt de caràcters qualssevol, és a dir, poden ser números, lletres o signes especials posats entre cometes. En anglès s'anomenen "string".

  Exemples: "hola" , "A27" , "IB245" , "262200"

- **Booleanes** o **lògiques**: serveixen per indicar el valor d'una expressió lògica. Només té els valors ver i fals (en anglès: true i false).

B) Una classificació un xic més completa en la que les dades es divideixen en dos grans tipus: **elemental i estructurat**.

** **Tipus elemental**

Les dades de tipus elemental són les més simples i consisteixen en un sol element.

Exemples: 3, B, cert, 5.7, dimarts

Aquests tipus solen estar definits en el propi llenguatge, o bé, es poden crear mitjançant l'enumeració dels valors que el formen.

Els tipus elementals més usats solen formar part de qualsevol llenguatge, és a dir, estan normalitzats i es coneixen amb el nom de **tipus estàndard**.

Els tipus estàndard són: enter, real, booleà i caràcter.

- Tipus **enter**:
  Es un subconjunt dels nombres enters. Hi ha un enter mínim i un enter màxim segons com els representa internament l'ordinador. Té les operacions suma, resta, producte i divisió entera, les quals són exactes. Cal controlar de no excedir els límits.
- Tipus **real**:
  Es un subconjunt dels nombres reals. Té les operacions suma, resta, producte i divisió, les quals no són exactes. Es la principal raó per distingir-los dels enters.
- Tipus **booleà**:
  Només posseeix els valors cert i fals. Té les operacions lògiques: i, o, no (en anglès: and, or, not). Els resultats s'obtenen amb les taules de veritat.
- Tipus **caràcter**:
  Es un conjunt de caràcters definits a l'equip de treball. El conjunt més normalitzat i usat és el codi ASCII, que té 224 caràcters imprimibles i 32 de control.

** **Tipus estructurat**

Les dades de tipus estructurat estan formades per dades elementals o predefinides agrupades de forma organitzada.

Exemples: les notes dels alumnes d'una classe, les dades personals d'un alumne, etc.

- **Vector i Matriu**

Es una repetició o iteració d'elements del mateix tipus. Per tant, el tipus matriu és una estructura homogènia.

Cada component és accessible de manera arbitrària o aleatòria mitjançant l'índex de selecció. Per indicar un element de la matriu hem de posar l'identificador i l'índex o selector.

El cas més senzill d'una estructura d'aquest tipus és una matriu d'una sola dimensió que s'anomena **vector**.

Intuïtivament, podem considerar un vector com un conjunt de calaixos amb el mateix nom i que només es diferencien per un número d'ordre anomenat índex que indica a quina dada de les del conjunt ens referim. Així, per indicar la posició k del vector A posarem A(k).

Igualment per referir-nos a un element d'una matriu posarem el seu nom seguit del valor dels índexs posats entre parèntesi.

A més de vector i matriu, cal citar, entre altres, aquests tipus estructurats: conjunt, llista, registre i fitxer.

3. Algorisme

3.1. Concepte d'algorisme

Un **algorisme** és una descripció que, de forma no ambigua, ens indica els passos que cal fer per tal de resoldre un problema.

És a dir, si algú coneix la manera de resoldre un problema, podrà escriure'n l'algorisme; aquest algorisme, comunicat a una altra persona, ha de permetre que aquesta també el pugui resoldre correctament i sense cap dubte.

La paraula algorisme fou inventada en honor del matemàtic àrab Al-Kuaritz-mi, que en el segle IX va escriure un llibre on, per primer cop, s'explicaven mètodes precisos per resoldre les quatre operacions aritmètiques elementals.

Veiem un parell d'exemples d'algorismes:

Exemple 1: Algorisme d'Euclides per calcular el màxim comú divisor (m.c.d.) de dos números enters i positius.

Pas 1: Demanar els dos números.

Pas 2: Prendre el número major com a divident i el menor com a divisor.

Pas 3: Trobar la resta de la divisió entera entre el divident i el divisor.

Pas 4: Si la resta és igual a zero, anar al pas 5; si no, prendre el divisor com a divident i la resta com a divisor i anar al pas 3.

Pas 5: El m.c.d. és el divisor de l'última divisió.

Pas 6: Fi.

Exemple 2: Algorisme per preparar un ou dur.

Pas 1: Agafar un ansat i posar-hi aigua.

Pas 2: Posar l'ansat al foc.

Pas 3: Esperar que l'aigua bulli.

Pas 4: Agafar un ou i posar-lo dins l'ansat.

Pas 5: Esperar 10 minuts.

Pas 6: Treure l'ansat del foc

Pas 7: Treure l'ou cuit de l'ansat.

Pas 8: Fi.

Com es pot veure, els algorismes no són domini exclusiu de la matemàtica o de la informàtica. En moltes accions de la nostra vida diària estem utilitzant de forma inconscient algorismes, sobretot en les accions més rutinàries. Així, per exemple, quan fem una truita a la francesa, quan seguim el manual d'instruccions d'un aparell electrònic, quan dibuixem, amb regle i compàs, un exàgon inscrit en una circumferència, etc.

Els algorismes poden ser expressats amb un conjunt de passos (o accions) que cal realitzar l'un a continuació de l'altre mentre no s'indiqui el contrari.

Un mateix problema pot tenir diferents mètodes de solució i, per tant, diferents algorismes, cada un d'ells amb avantatges i inconvenients. En aquests casos convé triar el que ens porti menys complicacions per a la seva programació. Per tant, davant d'un problema és convenient no quedar-se amb el primer algorisme que trobem sinó fer un esforç i intentar trobar-ne un de millor.

Per tal d'evitar les ambigüitats que existirien si aquestes "receptes de solució" les descrivíssim en llenguatge natural, s'utilitza una sintaxi pròpia i precisa que veurem a continuació.

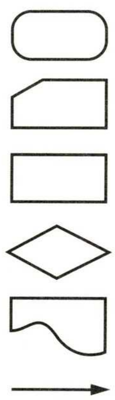
3.2. Mètodes d'anotació d'algorismes

Hi ha diversos mètodes per representar un algorisme: diagrames de flux, pseudocodi, diagrames de flux dimensional (DFD), etc. Aquí veurem els dos primers.

3.2.1. Diagrames de flux o ordinogrames

Un **diagrama de flux** (o **ordinograma**) és la representació gràfica d'un algorisme, mitjançant l'ús de signes convencionals subjectes a certes regles.

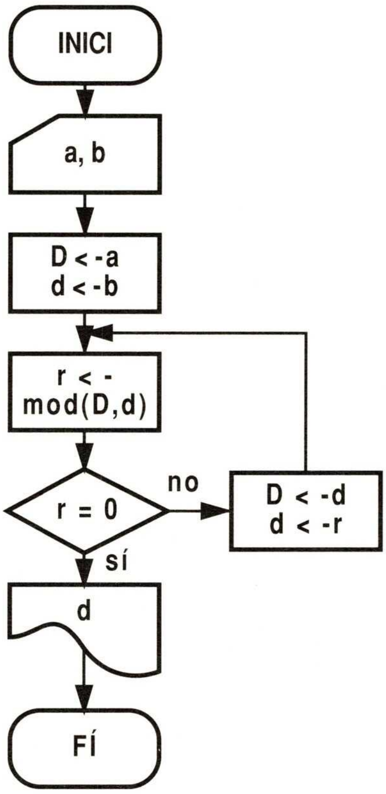
Els símbols que s'utilitzen són:



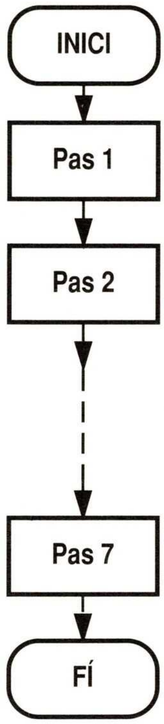
- Per indicar l'inici i la fi.
- Per indicar l'entrada de dades.
- Per indicar les operacions.
- Per indicar les condicions (bifurcacions).
- Per indicar la sortida de resultats.
- Línies orientades per enllaçar els símbols anteriors.

Exemples: Veiem el diagrama de flux dels exemples anteriors.

Algorisme d'Euclides



Preparació d'un ou dur



3.2.2. Notació algorísmica o pseudocodi

La **notació algorísmica** (o **pseudocodi**) és la descripció d'un algorisme mitjançant l'ús del llenguatge natural i d'unes estructures ja definides que veurem en el punt següent.

Exemple 3: Algorisme d'Euclides.

inici
    llegeix a,b
    D <- a
    d <- b
    r <- **resta_divisió_entera** (D,d)
    mentre r<>0 **fer**
        D <- d

```
                d <- r
                r <- resta_divisió_entera (D,d)
            fimentre
            escriu d
        fi
```

Exemple 4: Algorisme per trobar el més gran de tres números.

```
        inici
            llegeix a,b,c
            gros <- a
            si b > gros llavors gros <- b
            si c > gros llavors gros <- c
            escriu gros
        fi
```

3.3. Estructures algorísmiques bàsiques

Tal com hem vist a la definició d'algorisme, la resolució d'un problema comportarà una sèrie de passos o accions a fer. Una acció pot ser simple o composta.

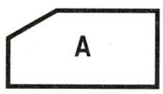
3.3.1. Instruccions d'acció simple

Recordem que en un sistema podem distingir tres fases: entrada, procés i sortida. D'altra banda, dins el procés, moltes vegades és necessari calcular una expressió i guardar el seu valor en una variable. Això dóna lloc als tres grans grups en què podem classificar les instruccions d'acció simple.

- **Instruccions de lectura (o d'entrada)**

Són aquelles que permeten assignar a una variable un valor procedent de l'exterior, és a dir, un valor a través del teclat o d'algún altre perifèric d'entrada.

En els diagrames de flux usarem el símbol amb el nom de la variable al seu interior.



En pseudocodi posarem el terme: **llegeix A**

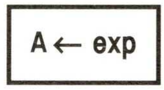
- **Instruccions d'assignació**

Són aquelles que serveixen per guardar en una variable el valor d'una constant, d'una altra variable o d'una expressió.

En aquest tipus d'instruccions convé distingir clarament:

* La variable a la qual s'assigna el valor.
* La constant o expressió, el valor de la qual és assignat a la variable.

En els diagrames de flux usarem el símbol

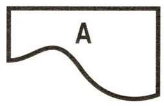


En pseudocodi es posa simplement: **A <- exp**.

- **Instruccions d'escriptura (o de sortida)**

Són aquelles que permeten conèixer els resultats obtinguts durant l'execució del procés.

En els diagrames de flux usarem el símbol amb el nom de la variable al seu interior.



En pseudocodi posarem el terme: **escriu A**

3.3.2. Estructures algorísmiques compostes

Hi ha tres estructures algorísmiques compostes bàsiques: seqüencial, condicional i iterativa.

- **Estructura seqüencial**

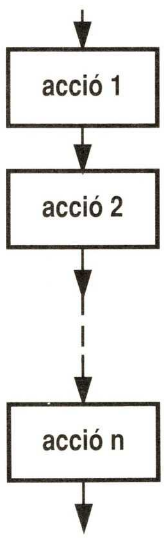
Consta d'una sèrie d'accions, acció 1 ... acció n, les quals han d'ésser realitzades una darrera l'altra, és a dir, començant per la primera i acabant per l'última.

Observeu que en una estructura seqüencial:

* S'executen totes les accions.
* Cada acció només s'executa una vegada.
* Les accions s'executen en l'ordre en què estan escrites.

Exemple: L'algorisme de l'exemple 2 té estructura seqüencial.

El diagrama de flux i la notació algorísmica de les estructures seqüencials són, respectivament:



inici
acció 1
acció 2
.
.
.
acció n
fi

- Estructura condicional

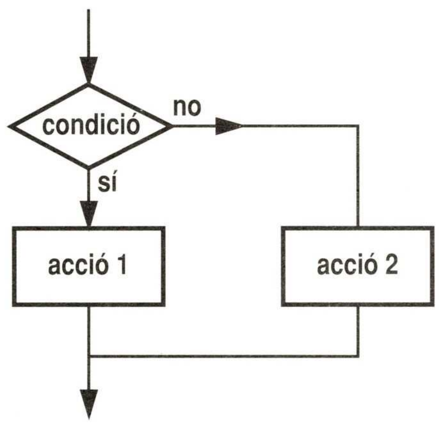
En veurem tres tipus. Els dos primers són els més directament relacionats amb el llenguatge natural. El tercer és una generalització del primer, implementat en llenguatges de programació estructurats com ara el Pascal.

Estructura condicional amb dues alternatives

Consta d'una condició i de dues accions, de les quals només se n'executarà una, la primera o la segona, segons que la condició sigui, respectivament, vera o falsa.

Exemple: Algorisme per escollir el preu d'una entrada al circ segons l'edat de la persona (un preu per als menors de 12 anys i un altre per a la resta de persones).

El diagrama de flux i la notació algorísmica són, respectivament:



si condició **llavors**
  acció 1
altrament
  acció 2
fisi

Estructura condicional amb una alternativa (simple)

Es pot considerar com una variant del primer tipus quan l'acció 2 és "no fer res". El diagrama de flux i la notació algorísmica són:



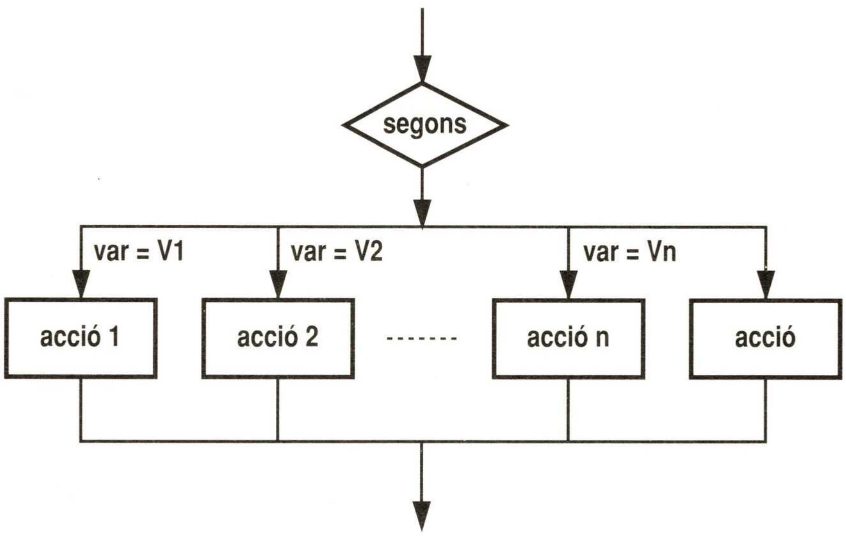
si condició **llavors**
  acció 1
fisi

Exemple: L'algorisme per saber si cal agafar el paraigües, segons plogui o no, té estructura condicional amb una alternativa.

Estructura condicional múltiple

Es una estructura condicional especial controlada per una variable. Aquesta variable pot prendre diferents valors i a cada un d'aquest valors li correspon una determinada acció.

El diagrama de flux és:



La notació algorísmica és:

segons_valgui variable
  v1: acció 1
  v2: acció 2
  ....
  vn: acció n
  altre: acció
fisegons

Estructura iterativa

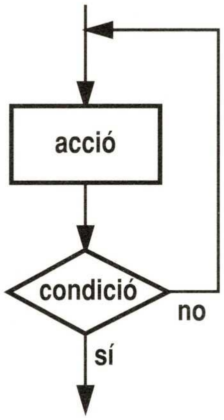
En veurem tres tipus. Els dos primers són els més directament relacionats amb el llenguatge natural. El tercer és una variant dels anteriors introduït pels primers llenguatges de programació d'alt nivell (Fortran, Basic, etc.).

Tipus **"fins que"**

Consta d'una condició i d'una acció que la precedeix, la qual es va repetint fins que la condició sigui vertadera.

Exemple: L'algorisme d'engegar un cotxe (donar el contacte fins que el motor s'engegui) té estructura d'aquest tipus.

El diagrama de flux i la notació algorísmica són:



repetir
acció
fins que condició

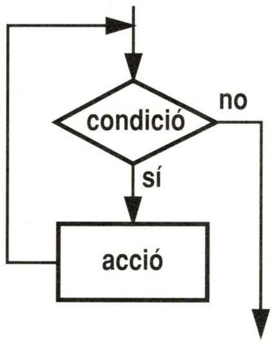
Noteu que, en aquest cas, l'acció s'executa al menys una vegada.

\# Tipus **"mentre"**

Consta d'una condició i d'una acció que la succeeix, la qual es va repetint mentre la condició sigui vertadera.

Exemple: L'algorisme de posar el segell en un piló de cartes (mentre en quedin) té aquest tipus d'estructura.

El diagrama de flux i la notació algorísmica són:



mentre condició **fer**
acció
fimentre

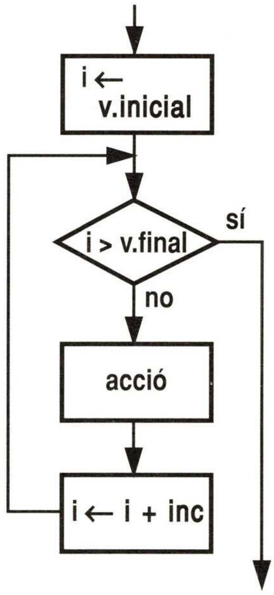
Noteu que, en aquest cas, l'acció pot no executar-se cap vegada.

\# Tipus **"indexat"**

Utilitza una variable anomenada índex i consisteix a repetir una acció des d'un valor inicial de l'índex fins a sobrepassar un determinat valor final; cada vegada que s'executa l'acció, es suma a l'índex un increment (positiu o negatiu).

Exemple: L'algorisme per calcular el quadrat dels números compresos entre 10 i 20, ambdós inclosos, té aquest tipus d'estructura.

El diagrama de flux és:



La notació algorísmica és:

per índex = v.inicial **fins a** v.final **incrementant** v.increment **fer**
acció
fiper

És evident que, en les estructures iteratives, l'acció ha de modificar els termes de la condició per tal que en algun momemt s'acabi la repetició.

4. Programació estructurada

4.1. Introducció

Cada cop es parla més de **programació estructurada**. De forma senzilla podríem dir que són els mètodes que permeten programar correctament. Això semblaria el mínim que es pot demanar en informàtica. Però programar correctament no és gens fàcil.

A l'apartat anterior hem vist els diagrames de flux, que permeten la descripció gràfica dels algorismes. Aquesta mena de llenguatge constitueix una primera eina d'ajuda al programador i així fou, històricament parlant. Ara bé, cal afegir que aquesta eina és útil per a problemes petits o mitjans, però no és pràctica per a problemes complexos.

Per això, aparegueren dos criteris de programació complementaris:

- Programació estructurada.
- Programació modular.

4.2. Programació estructurada

Alguns autors defineixen la **programació estructurada** com una forma d'elaboració de programes basada en els següents punts:

1.- Tot programa ha de ser escrit utilitzant únicament les estructures algorísmiques bàsiques.

2.- Disseny del programa usant la metodologia de l'Anàlisi Descendent.

El punt **1** fa referència al fet que tota estructura algorísmica es pot obtenir combinant adequadament les tres estructures bàsiques vistes (seqüencial, condicional i iterativa). El punt **2** es tracta més endavant.

El fet d'usar la programació estructurada comporta un nombre menor d'errors en el procés de programació i l'obtenció de programes que es poden llegir, entendre, modificar i codificar fàcilment pel seu autor o qualsevol altra persona que hagi d'utilitzar-lo.

Convé fer notar que d'altres autors consideren el punt 1 com la programació estructurada i el punt 2 com la programació modular.

Així, en atacar un problema es poden utilitzar criteris de programació modular per dividir-lo en parts independents i, després, utilitzar mètodes de programació estructurada a la programació de cada mòdul.

4.3. Anàlisi descendent

La metodologia d'**Anàlisi Descendent**, també coneguda amb el nom de "**top down**" o "**per refinaments successius**", va de les coses més generals a les més particulars.

Consisteix a descomposar el problema a tractar en subproblemes (que considerem com a accions). En un segon pas, podem agafar algunes d'aquestes accions i tractar-les com a problemes, descomposant-les en accions més simples, i això ho podem anar fent successivament fins a obtenir accions que ja no es poden descomposar més. És a dir, comencem escrivint un primer esquema de l'algorisme, deixant accions sense detallar; posteriorment, refinem aquestes accions, o sigui, les detallem; repetim aquest procés fins que no quedi cap acció per detallar.

Tot això es pot fer en llenguatge natural i, en tot cas, es pot usar la notació algorísmica o els diagrames de flux.

Exemple: Suposem que volem resoldre el problema de canviar la roda d'un cotxe.

En un primer pas podem indicar les següents accions:

1.- Parar i immobilitzar el cotxe.
2.- Agafar les eines i la roda de recanvi.
3.- Treure la roda rebentada.
4.- Posar la roda de recanvi.
5.- Guardar la roda rebentada i les eines.

Ara podem refinar algunes (o totes) d'aquestes accions. Per exemple, agafem l'acció 3 i la descomposem en altres accions més detallades:

3.1.- Afluixar els cargols.
3.2.- Aixecar el cotxe amb el gat.
3.3.- Treure els cargols.
3.4.- Retirar la roda del seu lloc.

Això fóra un segon pas de l'Anàlisi Descendent. I encara com a exemple d'un tercer pas, podem descomposar l'acció 3.2 en:

3.2.1.- Posar el gat sota el cotxe al lloc corresponent.
3.2.2.- Agafar la maneta i posar-la al gat.
3.2.3.- Voltar la maneta fins que obtinguem l'altura necessària.

5. Programació modular

5.1. Concepte de programació modular

Ja hem vist que la millor manera de resoldre un problema és dividir-lo en subproblemes per analitzar-los de manera separada.

La **programació modular** és un mètode de disseny consistent a dividir el problema total en parts perfectament diferenciades que poden ser analitzades, programades i acabades per separat.

Els objectius de la programació modular són:

- Disminuir la complexitat.
- Augmentar la claredat.
- Augmentar la fiabilitat.
- Facilitar la divisió del treball.
- Facilitar les modificacions i correccions.
- Obtenir recursos generalitzats.

5.2. Concepte de mòdul o subprograma

Un **mòdul** o **subprograma** és una part del programa format per un conjunt o bloc d'instruccions autònom, és a dir, que té una entitat pròpia i que serveix per realitzar una certa tasca ben determinada.

A diferència d'una estructura de control que només serveix per construir un programa, un mòdul o subprograma posseeix la seva pròpia arquitectura, les seves pròpies variables i, fins i tot, pot tenir d'altres subprogrames.

Podem definir el mòdul dient que:

- està format per una o diverses instruccions.
- es pot referenciar mitjançant un nom.
- es pot cridar des de diferents punts del programa.

Si una part del programa es repeteix en diversos llocs, és molt pràctic tractar-lo com un mòdul.

D'altra banda, hi ha mòduls o subprogrames que tenen un interès especial pel fet que són útils en diverses ocasions. Justament, la programació modular facilita l'elaboració de procediments generals que podran utilitzar-se en d'altres aplicacions.

Això dóna lloc al concepte de **biblioteca de subprogrames**, que és un conjunt de rutines o subprogrames, d'on es treuen aquelles que interessen en un moment determinat per tal d'afegir-les al programa principal.

Convé tenir molt clar i distingir perfectament la **declaració** d'un subprograma de la seva **crida**.

Declarar un subprograma consisteix a definir el seu funcionament, les dades que accepta, les tasques que realitza, i, si s'escau, els resultats que genera. Es tracta, doncs, d'una descripció.

La declaració d'un subprograma comprèn el seu nom, la llista dels paràmetres que se li han de passar, és a dir, les dades que espera en el moment de la seva crida, i la descripció de les accions que ha de fer.

Per la seva banda, **cridar** un subprograma vol dir activar-lo per tal que s'executi.

5.3. Procediments i funcions

Podem distingir dos tipus de subprogrames:

- Procediments (o subrutines)
- Funcions

Un **procediment** és un subprograma o mòdul que permet definir i executar una tasca determinada, més o menys complexa, mentre que una **funció** sempre retorna un resultat.

És a dir, el procediment realitza una acció i es correspon amb una extensió de la noció d'instrucció (es pot considerar com una superinstrucció). En canvi, la funció retorna un resultat i, en això, s'assembla a una expressió matemàtica.

Per exemple, si volem dibuixar un rectangle a la pantalla, això representa una acció i pot donar lloc a un procediment. En canvi, si volem calcular una fórmula matemàtica o determinar la posició que ocupa un cert caràcter en una cadena de caràcters, caldrà posar en marxa una funció.

De fet, i així ho fan llenguatges de programació com el Logo i el Pascal, es pot considerar una funció com un cas particular de procediment amb una única sortida. Una funció ha de tenir un lloc de l'algorisme on s'indica clarament el valor que retorna. A més, amb aquest valor s'ha de fer alguna cosa: assignar-lo, operar-lo amb d'altres valors, escriure'l, etc.

Exemple de funció:

```
funció mitjana(a,b)
inici
      retorna (a+b)/2
fi
```

Això seria la declaració o descripció de la funció. Per a la seva crida posaríem mitjana(30,12), en el cas de valors concrets, o bé, mitjana(x,y), suposant que x i y són variables del programa que crida aquesta funció.

Exemples:

```
c <- mitjana (30,12)
escriu mitjana(x,y)
```

5.4. Paràmetres d'un procediment

Un mòdul pot tenir entrades i sortides de dades. Les variables d'entrada i de sortida d'un procediment s'anomenen, respectivament, paràmetres d'entrada i de sortida.

Els **paràmetres d'entrada** són aquelles variables del procediment a les quals cal donar un valor abans d'ordenar-ne la seva execució.

Els **paràmetres de sortida** són aquelles variables a través de les quals el procediment retorna, al programa que l'ha cridat, el valor o valors que aquest necessitava en el moment de la crida.

Els paràmetres d'entrada d'una funció també es coneixen amb el nom d'**arguments** de la funció. Els arguments d'una funció no han de ser necessàriament valors numèrics, sinó qualsevol tipus de dada que processi el programa.

La manera de **cridar un procediment** es fa posant el seu nom seguit dels paràmetres entre parèntesi. Igualment, per fer referència a una funció escriurem el nom de la funció seguit dels arguments posats entre parèntesi.

* **Paràmetres formals i paràmetres reals:**

  Els **paràmetres formals** són les variables que s'utilitzen en el moment de declarar o definir el procediment mentre que els **paràmetres reals** són aquells valors o variables que s'usen en el moment de la crida.

Així, a l'exemple anterior, els paràmetres formals són a i b, i els paràmetres reals són x i y.

La correspondència entre paràmetres formals i reals s'estableix en funció de la seva posició a la llista.

*** Variables globals i variables locals:**

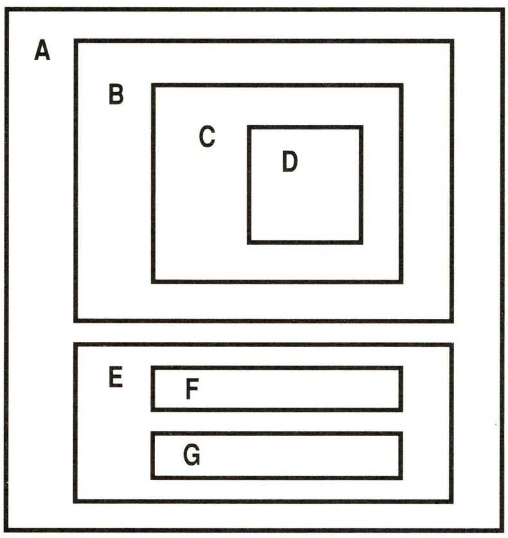
Un altre aspecte a tenir en compte en un procediment és el concepte de variables globals i variables locals.

- **variables globals**: són les variables definides a nivell més ampli que són utilitzades pel procediment.

- **variables locals**: són les variables específiques definides dins el cos del procediment, d'ús exclusiu d'aquest i que no tenen res a veure amb les variables del programa principal.

Les variables definides en un àmbit determinat són accessibles des d'aquest àmbit i des de qualsevol procediment interior.

Per exemple, si suposem l'esquema:



Variables definides a	Accessibles des de
A	A,B,C,D,E,F,G
B	B,C,D
C	C,D
D	D
E	E,F,G
F	F
G	G

5.5. Recursivitat

Un procediment és **recursiu** quan en algun lloc de la seva declaració es crida a si mateix. La recursivitat és una eina molt important i potent dins el món de la programació, que convé conèixer i utilitzar de forma adequada.

S'ha de vigilar a crear un procediment recursiu que no entri en un procés de repetició il.limitada. Per això hem d'assegurar-nos que en algun moment s'executa el procediment sense passar per les instruccions recursives.

Exemple de procediment recursiu:

```
procediment gira (paraula)
inici
        si paraula = "" llavors para
                      altrament gira (sense_primer(paraula))
        fisi
        escriu_seguit(primer(paraula))
fi
```

Pràctiques de Logo

Suggeriment de temporització de la Introducció a la Programació

SESSIÓ	TEORIA		PRÀCTICA
1	Met. Prog.: Capítol 1 Logo: Apartats 1, 2, 3, 4, 5, 6	Pl:	- Introducció al Logo. Conceptes generals. - Mode directe. Tipus de pantalla. - Definició i execució de procediments. - Edició de procediments.
2	Met. Prog.: Capítol 2 Logo: Apartats 7, 8, 9, 10, 11	P2:	- Exercicis sense ordinador - Fer càlculs amb l'ordinador - Pràctica amb Logo - Procediments amb paràmetres. - Procediments amb variables globals. - Gestió al Logo.
3	Met. Prog.: Capítol 3 Apartats: 3.1, 3.2, 3.3.1., 3.3.2 (part) Logo: Apartat 17	P4: P5:	- Algorismes "no programables" - Algorismes problemes senzills - Programes d'estructura seqüencial. - Instr. simples: entrada, sortida, assignació - Gràfics logo senzills.
4	Met. Prog.: Apartat 3.3.2 (part) Logo: Apartats 12, 13	P6:	- Exercicis expressions lògiques. - Problemes d'estructura condicional: numèrics. - Problemes d'estructura condicional: gràfics.
5	Met. Prog.: Apartat 3.3.2. (part) Logo: Apartat 14	P7:	- Problemes d'estructura iterativa. - Gráfiques amb logo. Les tres estructures.
6	Met. Prog.: Capitol 4 Logo: Apartat 18 (1a. part)	P8:	- Anàlisi Descendent. - Problemes globals senzills. - El món matemàtic a logo.
7	Met. Prog.: Apartats 5.1 i 5.2 Logo: Apartats 19, 20	P9:	- Modularitat - Problemes globals menys senzills. - Les llistes a logo.
8	Met. Prog.: Apartats 5.3 i 5.4 Logo: Apartats 15, 18 (2a. part)	P10:	- Funcions - El món matemàtic a logo.
9	Met. Prog.: Apartats 5.5 Logo: Apartat 16	P11:	- La recursivitat.
10	Met. Prog.: res Logo: Apartat 21	P12:	- Problemes globals més complexos. Projecte.
11		P12:	- Problemes globals més complexos. Projecte.
12		P12:	- Problemes globals més complexos. Projecte.

PRÀCTICA Nº 1

Introducció al Logo (I)

Objectius:

- Iniciar-se en el llenguatge Logo.
- Saber entrar i sortir de l'ambient Logo.
- Conèixer algunes instruccions per començar a treballar amb aquest llenguatge.
- Veure els diferents tipus de pantalla que té.
- Conèixer la forma com es defineixen i s'executen els procediments.
- Saber utilitzar l'Editor per crear o corregir procediments.

Exercicis:

1. Carregueu el llenguatge Logo a la memòria de l'ordinador.

2. Passeu a pantalla de text. Sempre que vulgueu podeu netejar aquesta pantalla amb la primitiva esborra.text.

3. Practiqueu amb la primitiva escriu, tot escrivint noms, frases o el resultat d'una expressió aritmètica.

Exemples: escriu "hola
escriu [Bon dia, Jordi]
escriu 4 * 6

4. Calculeu el resultat de les següents expressions:

237 + 5286 =
23 x 65 =
14 - 345 =
234 : 65 =

5. Comproveu què passa en donar la instrucció suma 4 7. El Logo ens dóna un missatge d'error. Recordeu que amb el resultat d'una operació sempre s'ha de fer alguna cosa. Com a mínim escriure'l a la pantalla. Així, escriu suma 4 7 funciona correctament. Això et recorda que hi ha dos tipus d'instruccions: ordres i operacions.

6. Fins ara has estat treballant en mode directe, és a dir, dónes una instrucció i en prémer <RETURN> aquella s'executa. Ara veurem com s'escriuen i s'executen els programes (procediments). Copia exactament el següent procediment:

```
procediment salutació
escriu [Hola, com et dius?]
posa.a "nom paraula.llegida
escriu [Benvingut al món del Logo!]
escriu frase [Espero que t'agradi] :nom
fi
```

Com has pogut comprovar, en acabar cada línia amb un <RETURN> no fa res a diferència del mode directe, excepte al final de tot que t'avisa que el procediment ha estat definit. Ara per executar-lo només cal escriure el seu nom. Prova-ho.

Pots fer-ho tantes vegades com vulguis. És una tasca que ha après i no es perd mentre tu no ho indiquis expressament o surtis del Logo.

7. Si t'has equivocat en escriure el procediment o bé vols canviar alguna cosa has d'anar a l'editor, que és una mena de processador de textos pensat per escriure o corregir procediments. Per anar-hi has d'escriure: edita "salutació

Ara corregeix o canvia alguna frase de les que presenta aquest primer procediment que has fet.

8. Copia, executa i després modifica a l'editor el següent procediment:

```
procediment sumes
escriu [Entra un número]
posa.a "num1 paraula.llegida
escriu [Entra un altre número]
posa.a "num2 paraula.llegida
escriu frase [La seva suma és] suma :num1 :num2
escriu [Vols fer-ne una altra (s/n)?]
posa.a "car car.llegit
si :car = "s [sumes]
fi
```

9. Fins ara hem treballat a la pantalla de text. Ara passarem a la pantalla mixta que ens permetrà fer gràfics (els coneguts gràfics de tortuga del Logo).

En primer lloc practicaràs en mode directe amb les primitives avança, recula, gira.dreta, gira.esquerra i inicia.dibuix.

Les quatre primeres han d'anar seguides d'un número que indica el valor del moviment o del gir. L'última serveix per netejar la pantalla gràfica i situar la tortuga al centre de la pantalla.

Exemples: avança 70
gira.dreta 30
inicia.dibuix

Algunes primitives del Logo es poden escriure de forma curta. En concret, les abreviatures d'aquestes cinc que estem tractant són: av, re, gd, ge i id.

10. En el món gràfic també podem definir procediments que més endavant executem. Copia i prova els següents procediments:

```
procediment casc
gd 90 av 80 gd 120 av 40 gd 60 av 120
gd 60 av 40 gd 120 av 80 ge 90
fi

procediment vela
av 100
gd 150
repeteix 3 [av 100 gd 120]
fi

procediment barca
id
desapareix
casc
vela
fi

procediment cuc
id
repeteix 5 [repeteix 6 [av 20 gd 90] gd 180]
fi
```

Observa que podem posar una instrucció en cada línia, o bé, diverses instruccions per línia separades per un espai. D'altra banda noteu com en l procediment barca fem servir la modularitat de Logo. Si voleu tornar a veure la tortuga heu d'usar la primitiva apareix.

Utilitza l'editor per fer alguna modificació en aquests procediments i torna'ls a executar per veure els canvis.

11. Ara guardarem aquests 6 procediments de la memòria al disquet. Posa el teu disquet de treball a la unitat A.

Escriu: desa “pract1 procs i prem <RETURN>.

12. Finalment sortirem de l’ambient Logo per tornar al sistema operatiu. Això s’aconsegueix amb la primitiva .adéu.

PRÀCTICA Nº 2

Dades

Objectius:

- Saber escriure correctament les constants numèriques.
- Repassar l'ordre de prioritat dels operadors matemàtics.
- Saber escriure correctament les expressions matemàtiques tant des del punt de vista matemàtic com informàtic.
- Saber utilitzar el Logo per al càlcul d'expressions matemàtiques.

Exercicis:

1. Assenyaleu quines de les següents constants numèriques són incorrectes:

1400 45 1.328,25 2.4E7 3.86 7E-3 .25

2. Escriviu en notació normal les següents constants numèriques:

2.6E4 1.23E-2 5E5 -2.34E-6

3. Escriviu en notació exponencial o científica les següents constants numèriques:

18400 0.079 1234567 .00000025

4. Calculeu el valor de les següent expressions:

a) 4+7x2 = **b)** $6+2^3+8:2$ = **c)** -7+6x4:3 =

d) (4-2)x5+3x2 = **e)** 2x(3+6)-6:3 = **f)** $4+(2-8)^2$ =

5. Escriviu les expressions de l'exercici anterior en forma correcta per a un ordinador.

6. Assenyaleu quines de les expressions següents són, des d'un punt de vista informàtic, incorrectes:

a) 6-2(3+5) **b)** A^2+5.B **c)** preu/2+C

d) (A.V-c^2)*5 **e)** 6*I+A **f)** sou*0,20

7. Corregiu les expressions incorrectes de l'exercici anterior.

8. Escriviu en forma "informàtica" les següents expressions:

a) $\frac{m+n}{2}$ **b)** $2(x+y-z) - \frac{k}{4}$ **c)** $\frac{-b}{a}$

d) $4\pi r^2$ **e)** $5x^3+7x^2-x+3$ **f)** $\frac{12x}{y+z}$

9. Calculeu, utilitzant el llenguatge Logo, el valor de les següents expressions:

a) 7+3x5 =

b) $12+2^4+28:7$ =

c) -4+2x9:5 =

d) (6-9)x3+7x8 =

e) 4x(2+8)-6:4 =

f) $7+(3-8)^2$ =

PRÀCTICA Nº 3

Introducció al Logo (II)

Objectius:

- Comprendre el concepte de variable i veure la diferència entre el nom de la variable i el seu contingut.
- Entendre i saber treballar amb els procediments amb paràmetres.
- Conèixer les instruccions més elementals per a l'entrada i la sortida de dades.
- Conèixer algunes primitives per a la gestió de l'espai de treball de l'ambient Logo.
- Conèixer la manera de guardar i recuperar el treball efectuat amb Logo.

Exercicis:

1. Treballant en mode directe creeu i inicialitzeu diverses variables amb la primitiva posa.a:

- Omple la variable "quantitat amb el valor 20.
- Omple la variable "preu amb el valor 1250.
- Comprova amb la primitiva escriu que cada una d'aquestes variables conté el valor que li has assignat.
- Omple la variable "total amb el producte de les dues anteriors.
- Mira quin valor conté la variable "total.
- Omple la variable "nom amb el teu nom.
- Omple la variable "cognom amb el teu primer cognom.
- Dóna la següent instrucció:
  posa.a "nom.complet frase :nom :cognom
- Quin és el contingut de la variable "nom.complet
- Inventa't algunes variables i assigna'ls objectes (números, paraules o llistes). Després comprova el seu contingut amb la primitiva escriu.

2. En aquest punt es tracta de veure que de la mateixa forma que hi ha primitives que necessiten d'un o més valors per funcionar correctament, nosaltres podem definir procediments amb paràmetres, és a dir, procediments

que a l'hora d'executar-se han d'anar seguits de tants valors com paràmetres tenen en la seva definició.

Proveu primitives com ara: avança, escriu, suma, etc. sense posar res al darrere i veureu com el Logo us dóna un missatge d'error.

Copieu i comproveu els següents procediments. Recordeu que si us equivoqueu podeu utilitzar l'editor per corregir-los.

```
procediment retalla :p
si :p = " [acaba]
escriu :p
retalla sense.últim :p
fi
```

```
procediment espiral :pas :angle
av :pas
gd :angle
espiral :pas + 5 :angle
fi
```

Ara per executar aquests procediments s'ha d'escriure el seu nom seguit d'una paraula en el primer cas i de dos números adients en el segon cas. Per aturar el segon procediment has de prémer la tecla F10. Per exemple, prova a fer:

```
retalla "papallona
```

```
espiral 10 90
```

Pots executar els dos procediments amb d'altres valors.

3. Hem vist que per entrar dades es pot fer mitjançant paràmetres. A vegades aquest mètode no és el més idoni. Llavors hem d'utilitzar la primitiva paraula.llegida, la qual atura el programa fins que l'usuari entra un valor numèric o alfanumèric. Aquest valor s'ha d'assignar a una variable.

D'altra banda, per obtenir els resultats d'un procés, hem de fer servir la primitiva escriu que ja coneixem d'exemples anteriors.

Per tal de comprendre millor aquestes dues primitives copia, executa i reflexiona sobre el següent procediment.

```
procediment àrea.rectangle
escriu [Entra la base]
posa.a "base paraula.llegida
```

```
escriu [Entra l'altura]
posa.a "altura paraula.llegida
posa.a "àrea :base * :altura
escriu frase [L'àrea del rectangle és] :àrea
fi
```

4. Recorda que tot el que fem amb l'ordinador s'ha de gravar al disquet si volem guardar-ho per més endavant, ja que en apagar l'ordinador o, fins i tot, en sortir del Logo i anar al sistema operatiu, es perd tot el que hi ha a l'espai de treball.

Grava, doncs, en el disquet, la feina feta fins ara amb la instrucció:

```
desa "pract3 [retalla espiral àrea.rectangle]
```

o bé,

```
desa "pract3 procs
```

si vols guardar tots els procediments de la memòria.

L'operació inversa, és a dir, recuperar del disquet les coses que hi tenim gravades es fa amb la primitiva recupera.

Per provar-ho recupera el treball de la pràctica 1 tot escrivint:

```
recupera "pract1
```

5. En aquest punt veurem algunes primitives per a la gestió de l'espai de treball. Concretament hauries de provar:

```
títols?
```

Observa com els procediments de la pràctica 1 estan a l'espai de treball, ja que formen part de la llista que ens dóna la primitiva títols. Ara, si volguessis, els podries utilitzar.

```
presenta [salutació cuc]
```

Et permet veure la definició dels procediments que inclous dins la llista, encara que aquí no els pots corregir.

```
oblida [retalla espiral]
```

Torna a fer:

```
títols?
```

Evidentment, els dos procediments que has manat esborrar (retalla i espiral) ja no estan a l'espai de treball.

Torna'ls a recuperar del disquet. Aquest cop ja no t'ajudo. Cal que et recordis de com es fa. Ho hem vist anteriorment.

Per acabar, practica una estona amb aquestes primitives per a la gestió de l'espai de treball i dels arxius de procediments.

Finalment surts del Logo amb la instrucció corresponent.

PRÀCTICA Nº 4

Algorismes i l'estructura seqüencial

Objectius:

- Entendre el concepte d'algorisme i veure que hi ha diversos mètodes per a la seva anotació.
- Veure que el concepte d'algorisme no és purament matemàtic o informàtic treballant algorismes no programables.
- Conèixer i saber aplicar les instruccions algorísmiques d'acció simple i l'estructura algorísmica seqüencial.
- Saber escriure l'algorisme de problemes senzills.

Exercicis:

1. Suposem que en un algorisme intervenen les variables a, b i c.

Indica en cada cas quin valor tindrà la variable c, després d'executar les línies següents:

a <- 10	a <- 10	a <- 8	a <- 8	a <- 3*(-2)
b <- 20	b <- a+5	b <- 5	b <- a-5	b <- a+4
c <- a-b	c <- a*b	c <- 20	c <- a*b	c <- b+a
		c <- a+b	c <- a+c	
c =	c =	c =	c =	c =

2. Quina assignació hi haurà d'haver en un algorisme que calculi:

a) La longitud d'una circumferència a partir del radi.
b) La mitjana aritmètica de tres números.

3. Escriviu en forma d'instrucció d'assignació:

a) $$x \leftarrow \frac{-b \pm \ b^2-4ac}{2a}$$

b) $h <- 5 + \dfrac{(x-5)^2}{y}$

4. Escriviu les línies d'un algorisme que permetin intercanviar el valor de dues variables A i B, és a dir, posar el valor d'A a B i el de B a A.

5. Descriviu, mitjançant el llenguatge natural, l'algorisme per resoldre cada un dels següents problemes:

a) Canviar una bombeta fosa.
b) Preparar un plat d'arròs bullit.
c) Arranjar la roda rebentada d'una bicicleta.
d) Escoltar una cinta de música en una cassette.
e) Dibuixar en un foli una circumferència de radi 10 cms i centrada en el punt mig del paper.
f) Traçar la recta mediatriu a un segment AB dibuixat en un foli.
g) Traçar una circumferència que passi per tres punts donats no alineats.
h) Trobar el centre d'una circumferència dibuixada en un paper.

6. Descriviu, mitjançant els diagrames de flux o la notació algorísmica, l'algorisme de resolució dels següents problemes:

a) Donada la longitud del costat d'un quadrat, calcular el seu perímetre.
b) Donada la longitud del radi d'una circumferència, calcular l'àrea del cercle comprès.
c) Calcular la mitjana aritmètica de tres números.
d) Donada una quantitat i un % de descompte, calcular el valor del descompte absolut i la quantitat a pagar.
e) Repartir una herència de forma directament proporcional a les edats de tres persones.
f) Donada l'aresta d'un cub, calcular la seva àrea i el seu volum.
g) Donats els costats d'un rectangle, calcular el seu perímetre, la seva àrea i la longitud de la diagonal.
h) Donada una quantitat en dòlars i el canvi en què es troba, calcular quantes pessetes són.

PRÀCTICA Nº 5

Algorismes amb estructura seqüencial

Objectius:

- Conèixer i saber aplicar les instruccions algorísmiques d'acció simple i l'estructura algorísmica seqüencial.
- Saber codificar en un llenguatge de programació els algorismes de problemes senzills.

Exercicis:

1. Si en algun moment d'un programa cal escriure a la pantalla les variables A, B, C, D, E i F, redacteu les ordres que calen perquè:

a) Escrigui una variable a cada línia.
b) Escrigui dues variables a cada línia.
c) Escrigui les sis variables en una línia.

2. Elaboreu els següents programes:

a) Que entrant el radi de la circumferència calculi la seva longitud i l'escrigui.
b) Que entrant la base i l'altura d'un triangle calculi la seva àrea i escrigui la base, l'altura i l'àrea.
c) Que calculi la hipotenusa d'un triangle rectangle havent entrat els dos catets.
d) Que passi a segons un temps entrat en hores, minuts i segons.

3. Escriviu un programa que permeti el càlcul de l'àrea d'un rectangle donades la base i l'altura:

a) amb instruccions d'entrada de la base i altura.
b) amb dos paràmetres.

4. Escriviu un bloc de programa que canviï el valor de dues variables A i B, és a dir, posi el valor d'A a B i el de B a A. Afegiu les instruccions necessàries per configurar un petit programa que permeti la seva comprovació.

5. Codifiqueu i comproveu els algorismes programables de l'exercici 6 de la pràctica 4.

6. Escriviu un procediment que entrant el nom d'una persona escrigui a la pantalla un missatge del tipus:

****

Hola, Andreu. Benvingut al món del Logo

****

7. Milloreu el programa anterior de tal manera que el missatge aparegui més o menys centrat en una pantalla neta. Haureu d'utilitzar les primitives: escriu, mou.cursor i esborra.text.

8. Dissenyeu, fent servir només l'estructura seqüencial, els procediments per dibuixar les figures indicades a continuació:

a) Quadrat de 50 unitats de costat.
b) Rectangle de 100 unitats de base i 40 d'altura.
c) Una creu.
d) Una aspa.
e) Una L.
f) Una casa senzilla.
g) Triangle equilàter de 80 unitats de costat.
h) Un asterisc.

Nota: En les figures en què no s'indica la mida hi poseu la que considereu idònia.

PRÀCTICA Nº 6

Algorismes amb estructura condicional

Objectius:
- Conèixer i saber calcular expressions lògiques.
- Saber aplicar l'estructura algorísmica condicional per a la resolució de problemes.

Exercicis:

1. Suposant que a=5, b=7 i c=2, calculeu el valor de les següents expressions lògiques:

a) a<b **b)** c<=6 **c)** 2*a>c

d) a<10 i 7>c **e)** a<b o c=5 **f)** no a>6

2. Descriviu de forma estructurada el següent algorisme per anar a la feina:

Pas 1: Anar a l'aparcament a buscar el cotxe.
Pas 2: Si el cotxe s'engega, anar al pas 3, si no, anar al pas 4.
Pas 3: Anar al lloc de treball amb el cotxe. Anar al pas 7.
Pas 4: Si tenim pressa, anar al pas 5, si no anar al pas 6.
Pas 5: Buscar un taxi perquè ens porti al lloc de treball. Anar al pas 7.
Pas 6: Agafar un autobús que ens porti a la feina.
Pas 7: Fi.

3. Trobeu l'algorisme per resoldre cada una de les situacions següents:

a) Buscar el més gros de dos números donats.
b) Buscar el més gros de tres números donats.
c) Buscar el més petit de tres números donats.
d) Donats dos números naturals diferents de zero esbrinar si un d'ells és divisor de l'altre.
e) Donades les amplituds de tres angles esbrinar si poden ser o no els tres angles d'un triangle.
f) Classificar un triangle en equilàter, isòsceles o escalè a partir de les longituds dels seus costats.

g) Classificar un triangle en acutangle, rectangle o obtusangle a partir de les amplituds dels seus angles.

h) Ordenar en sentit creixent tres números donats.

4. A l'oficina de facturació d'un magatzem apliquen un 20 % de descompte a totes les compres que superin les 50000 ptes. A més s'ha d'aplicar un recàrrec d'un 6_% sobre l'import net de la factura en concepte d'IVA. Elaboreu l'algorisme que donat l'import brut d'una compra ens proporcioni la quantitat total a pagar després d'aplicar el descompte (si s'escau) i el recàrrec.

5. Modifiqueu l'algorisme de l'exercici anterior per tal que també s'apliqui un descompte d'un 5_% a les compres inferiors a 50000 ptes.

6. Dissenyeu un algorisme que transformi una qualificació numèrica en qualitativa seguint la relació:

nota<3 .... Md
3<=nota<5 .... Ins
5<=nota<6 .... Suf
6<=nota<7 .... Bé
7<=nota<8.5 .... Not
8.5<=nota .... Exc

7. Escriviu els programes corresponents als algorismes de l'exercici 3 i comproveu la seva efectivitat.

8. Escriviu i comproveu els següents programes:

a) Programa que compari dos números entrats per l'operador i que respongui si són iguals o quin d'ells és el més gros.

b) Programa que ordeni:

b.1) dos números.

b.2) tres números.

PRÀCTICA Nº 7

Algorismes amb estructura iterativa

Objectius:

- Saber aplicar l'estructura algorísmica iterativa o repetitiva per a la resolució de problemes.
- Aplicar les diferents estructures per a la resolució de problemes de caire gràfic a l'ambient dels Gràfics de Tortuga.

Exercicis:

1. Feu un programa que:

a) Escrigui a la pantalla la taula de multiplicar del 5.

b) Escrigui les taules de multiplicar de l'1 al 10 de manera que cada taula quedi escrita a la pantalla durant un cert temps, després del qual s'esborri la pantalla i aparegui la taula següent.

c) Presenti a la pantalla la taula de multiplicar del número que l'usuari entra com a dada.

d) Calculi l'àrea de 6 rectangles entrant de forma successiva el valor dels seus costats.

e) Calculi la mitjana aritmètica d'una sèrie de valors. Per indicar el final de la sèrie s'ha d'entrar un valor especial que no compta.

f) Que compari, successivament, números que l'operador va entrant i que escrigui a la pantalla si són més petits que 5, més grossos que 5 o iguals a 5. Igual com abans, per acabar, entreu un número especial que no compta.

g) Que entrant, de forma successiva, els dos catets d'un triangle rectangle ens calculi el valor de la hipotenusa. Feu que el programa pari en entrar un catet igual a 0.

h) Que escrigui a la pantalla tots els números naturals parells inferiors a un número entrat per l'operador.

i) Que entrant, successivament, diferents números digui al final quants n'hi ha més petits que 50, quants entre 50 i 100 i quants més grossos que 100. Utilitzeu un número especial per indicar el final de la sèrie de números.

j) Que calculi la suma dels n primers números naturals.

k) Que calculi la suma dels 100 primers números parells.

l) Que calculi la suma dels números senars menors que un número entrat per l'usuari

2. Dissenyeu, utilitzant l'estructura iterativa, els procediments per dibuixar les figures indicades a continuació:

a) Quadrat de 50 unitats de costat.
b) Rectangle de 100 unitats de base i 40 d'altura.
c) Una creu.
d) Una aspa.
e) Triangle equilàter de 80 unitats de costat.
f) Un asterisc.
g) Deu línies paral.leles i verticals de 60 unitats de longitud i separades entre si, 10 unitats.
h) Les ratlles que indiquen els números d'un rellotge i una agulla que es mogui.
i) Una finestra formada a partir de quatre quadrats.
j) Sis quadrats petits posats en columna.

PRÀCTICA Nº 8

Resolució de problemes (I)

Objectius:
- Conèixer i saber aplicar la metodologia de l'anàlisi descendent per a la resolució de problemes.
- Saber resoldre qualsevol problema de mitjana dificultat tot aplicant les tres estructures algorísmiques estudiades.

Exercicis:

1. Dissenyeu un algorisme per:

a) Encendre una estufa de butà.
b) Rentar roba amb una rentadora automàtica.
c) Fer una truita de patates.
d) Sumar dos trencats.
e) Ordenar, alfabèticament, tres noms.
f) Parlar per telèfon amb una persona el número de la qual no sabem.
g) Comptabilitzar els resultats d'un partit de tennis.
h) Ordenar, alfabèticament, segons l'autor, una col.lecció de 50 llibres.

Nota: Apliqueu, quan sigui convenient, la metodologia de l'anàlisi descendent.

2. Descriviu l'algorisme de resolució dels següents problemes i, després, escriviu i executeu els programes corresponents:

a) Escriure 100 vegades la frase "Sóc el millor".

b) Escriure els 100 primers números parells:
   b.1) utilitzant la primitiva repeteix.
   b.2) utilitzant recursivitat de cua.

c) Generar una taula dels quadrats de l'1 al 20 de la forma

Taula de quadrats

1	1
2	2
....	
20	400

d) Mirar les monedes necessàries per pagar una quantitat inferior a 1000 ptes.

e) Trobar quin és el més gros d'un conjunt de números:

e.1) Sabent de bon principi quants números hi ha.

e.2) Posant un número especial que no compta i que serveix per indicar que s'han acabat.

3. Elaboreu un programa que calculi, a petició de l'usuari, una de les incògnites de la fórmula de l'interès simple havent entrat les altres tres. La fórmula és:

$$i = \frac{c\,r\,t}{100}$$

4. Escriviu un programa per discutir i resoldre l'equació de segon grau $ax^2 + bx + c = 0$, entrant com a dades els tres coeficients a, b i c.

5. Dissenyeu un algorisme que calculi l'àrea d'un nombre desconegut de rectangles de tal manera que el programa pari quan:

a) Es respon negativament a una pregunta que es formula a l'operador.

b) S'entra una dada impossible com a valor d'un costat (per exemple, un zero o un número negatiu).

c) S'entra un valor, prèviament, convingut.

6. Construïu els procediments amb paràmetres per dibuixar:

a) Quadrat de costat variable.

b) Rectangle de costats variables.

c) Triangle equilàter de costat variable.

d) Una L feta amb rectangles estrets.

e) Un molinet de triangles equilàters.

f) Un pentàgon de costat variable.

g) Un polígon de mida i número de costats variables.

h) Una circumferència de mida variable.

i) Un arc de circumferència de mida variable.

j) Una flor amb un nombre variable de pètals.

PRÀCTICA Nº 9

Resolució de problemes (II)

Objectius:

- Entendre i aplicar el concepte de la programació modular per a la resolució dels problemes.
- Saber resoldre qualsevol problema de mitjana dificultat tot aplicant les tres estructures algorísmiques estudiades.
- Saber utilitzar les llistes de forma àgil i correcta en la resolució dels problemes que ho requereixin.

Exercicis:

1. Construïu, aplicant el concepte de modularitat, els procediments amb paràmetres per dibuixar:

a) Una casa utilitzant els procediments quadrat i triangle.

b) Els dibuixos proposats al full annex.

c) Una branca senzilla.

d) Floc de neu.

e) Una filera de branques.

f) Una bandera.

g) Una roda de banderes.

h) Una filera de banderes.

2. Trobeu l'algorisme i, després, escriviu el programa per resoldre els següents problemes:

a) Escriure en pantalla el rètol continu:
.logo.logo.logo.logo.logo...

b) Assignant el nom i els cognoms a una llista, treure els tres mots separadament, un a cada línia.

c) Donada una llista de dos elements (nom i cognom), s'han d'escriure a la mateixa línia al revés (cognom i nom) amb una coma intercalada.

d) Escriure els elements d'una llista en columna.

e) Escriure els caràcters d'una paraula en columna.
f) Escriure a la pantalla una llista en forma decreixent.
g) Escriure a la pantalla una paraula en forma decreixent.
h) Dibuixar una línia d'asteriscs de longitud variable:
   h.1) horitzontal.
   h.2) vertical.

3. Trobeu l'algorisme i, després, escriviu el programa per resoldre els següents problemes:

a) Dir si una paraula conté alguna a.
b) Comptar el nombre d'"a" que conté una paraula qualsevol.
c) Esbrinar si una paraula qualsevol conté el diftong "ai"
d) Mirar si una paraula comença i acaba amb la mateixa lletra.
e) Comptar el nombre de vocals i consonants que té una paraula.
f) Donada una llista de paraules amb un nombre qualsevol d'elements, escriure l'última lletra de cada paraula.
   f.1) Aplicant recursivitat.
   f.2) Utilitzant les primitives element i n.elements.

4. Escriviu un programa que guardi una colla de números que l'usuari entra pel teclat en una llista i després calculi la suma de tots els elements de la llista.

5. Penseu l'algorisme i escriviu el programa que transformi una data de la forma 12-05-90 a 12 de maig de 1990. Suposem que la data sempre s'escriu amb 8 caràcters, és a dir, que quan el número del dia o del mes és inferior a 10 s'afegeix un zero al davant.

6. Escriviu un programa que demani dia, mes i any i ho presenti en la forma dd/mm/aa.

7. Elaboreu un programa que classifiqui una colla de números naturals que l'usuari va entrant pel teclat en tres grups (llistes): menors de 50, entre 50 i 100 (ambdós inclosos) i majors de 100.

8. Dissenyeu un programa que escrigui una taula del codi ASCII a partir del caràcter 32. En una columna el codi i a l'altra el caràcter corresponent.

9. Elaboreu un programa que, entrant una xifra de l'1 al 9, l'ordinador respongui el nom de la xifra.

10. Elaboreu un programa que escrigui, en lletres, un número inferior a 1000 entrat per l'usuari.

11. Dissenyeu un programa per mirar quants bitllets i monedes es necessiten per pagar una quantitat donada.

12. Penseu i escriviu un programa per calcular la recaptació setmanal de vendes d'una gasolinera que distribueix dos tipus de gasolina i un de gas-oil. Hom disposa, diàriament, al llarg de tota la setmana, de les quantitats distribuïdes de cada tipus de carburant. També es coneix el preu de cada un d'ells.

PRÀCTICA Nº 10

Funcions

Objectius:

- Conèixer i saber utilitzar les funcions matemàtiques que, en general, tenen tots els llenguatges de programació.
- Saber definir i utilitzar els procediments de tipus funció creats per un mateix.

Exercicis:

1. Escriviu en logo les següents expressions matemàtiques:

a) $\dfrac{2\ \text{tg}\ x}{1+\text{tg}^2(x/2)}$

b) $\dfrac{|A - B|}{2}$

c) $(\cos^2 x + \sin^2 y)^{1/2}$

d) $4(e^x + e^y)$

e) $\dfrac{\cos^2(x/2) - \sin^2(x/2)}{\cos x + \sin x}$

f) $\ln x + \ln (x^2+4x-5)$

2. Construïu una funció que retorni el valor absolut d'un número.

3. Construïu una funció que retorni el doble d'un número.

4. Elaboreu un programa que imprimeixi la taula d'arrels dels 100 primers números.

5. Escriure una taula de sinus encolumnada. Els angles han de variar des de 0 a 90 graus, de 5 en 5 graus. Utilitzeu la primitiva mou.cursor per aconseguir que les columnes quedin ben alineades.

6. Elaboreu un programa que faci unes taules trigonomètriques dels angles entre 0 i 90 graus, de grau en grau, de la forma:

graus   sinus   cosinus   tangent   cotangent

7. Elaboreu un programa que simuli 20 tirades d'un dau.

8. Construïu una funció que, donat un número, retorni ver si és parell i fals si és senar.

9. Construïu una funció tal que, donada una paraula retorni, el número de vocals que conté.

10. Construïu una funció tal que, donada una paraula en minúscula, la retorni en majúscula.

11. Escriviu un programa que calculi el quadrat, el cub, l'arrel quadrada i l'arrel cúbica de tots els números compresos entre dos números que entra l'operador.

12. Dissenyeu un programa que calculi l'àrea d'una corona circular a partir d'una subrutina (funció) que calcula l'àrea d'un cercle.

13. Copieu els següents procediments:

```
proc demo :n
escriu :n
canvia :n
escriu :n
fi
```

```
proc canvia :n
posa.a "n :n+1
escriu :n
fi
```

Executeu, per exemple, demo 1.

Reflexioneu sobre els resultats que s'obtenen. Us ha de servir per entendre millor el concepte de variable local. Recordeu que els paràmetres d'un procediment són variables locals.

14. Representeu, utilitzant la primitiva punt, la gràfica de la funció y=2x+4 a l'intèrval [-5,5].

15. Representeu la gràfica de la funció $y=x^2-4x+3$ a l'intèrval [-2,6].

16. Representeu la gràfica de la funció trigonomètrica y = sin x a l'intèrval [-8,8].

8. Ara és el contrari. Tot i que el Logo ja té les primitives n.elements, pertany i element, us proposem de construir unes funcions que facin la mateixa tasca, és a dir:

a) Funció que retorni el número d'elements d'una llista.

b) Funció que comprovi si un element pertany a una llista.

c) Funció que retorni l'element n-èssim d'una llista.

9. Construïu una funció que retorni el valor màxim dels continguts en una llista.

10. Dissenyeu un algorisme per resoldre el problema de les Torres d'Hanoi. Es tracta de passar una pila de discos de mida decreixent d'una posició A a una altra C, disposant d'una posició de treball B i seguint estrictament aquestes dues regles:

a) Només es pot portar un disc en cada moviment.
b) No es pot posar un disc sobre un altre de mida inferior.

PRÀCTICA Nº 12

Resolució de problemes (III)

Objectius:

- Repassar i consolidar la utilització dels diversos recursos estudiats per a la resolució informàtica de problemes.
- Agafar experiència i desimboltura en la resolució de problemes d'una certa dificultat.

Exercicis:

A) Diversos: Dissenyeu l'algorisme i escriviu el corresponent programa per a la resolució dels següents problemes de caire divers:

1. Conjugar el present d'indicatiu d'un verb regular de la primera conjugació entrat per l'usuari.

2. Generar 12 sumes aleatòries de dos números menors que 100. Les sumes han d'aparèixer a la pantalla en forma de taula i han d'estar distribuïdes en 4 fileres de 3 sumes cada una.

3. Presentar, centrades a la pantalla, 20 notes musicals (do, re, mi, fa, sol, la, si) escrites a l'atzar i una a continuació de l'altra.

4. Escriure una llista indefinida d'entrepans diferents formats aleatòriament a partir d'una llista de farcits i una altra de verdures.

Ex.: entrepà de pernil amb enciam
entrepà de formatge amb tomàquet
....

5. Dibuixar una línia diagonal amb un número variable d'asteriscos. Feu variables també la fila i la columna a partir de la qual es dibuixa. Ha de ser, per tant, un procediment amb tres paràmetres.

6. Escriure la taula de multiplicar del número que entra l'operador, procurant que els números quedin ben alineats.

7. Mirar si un número o una paraula és o no cap-i-cua. S'enténque un número o una paraula és cap-i-cua quan llegit del dret o del revés és el mateix.

8. Donat el dia i el mes del naixement d'una persona, esbrinar el seu signe de zodíac.

8. Ara és el contrari. Tot i que el Logo ja té les primitives n.elements, pertany i element, us proposem de construir unes funcions que facin la mateixa tasca, és a dir:

a) Funció que retorni el número d'elements d'una llista.

b) Funció que comprovi si un element pertany a una llista.

c) Funció que retorni l'element n-èssim d'una llista.

9. Construïu una funció que retorni el valor màxim dels continguts en una llista.

10. Dissenyeu un algorisme per resoldre el problema de les Torres d'Hanoi. Es tracta de passar una pila de discos de mida decreixent d'una posició A a una altra C, disposant d'una posició de treball B i seguint estrictament aquestes dues regles:

a) Només es pot portar un disc en cada moviment.
b) No es pot posar un disc sobre un altre de mida inferior.

PRÀCTICA Nº 12

Resolució de problemes (III)

Objectius:

- Repassar i consolidar la utilització dels diversos recursos estudiats per a la resolució informàtica de problemes.
- Agafar experiència i desimboltura en la resolució de problemes d'una certa dificultat.

Exercicis:

A) Diversos: Dissenyeu l'algorisme i escriviu el corresponent programa per a la resolució dels següents problemes de caire divers:

1. Conjugar el present d'indicatiu d'un verb regular de la primera conjugació entrat per l'usuari.

2. Generar 12 sumes aleatòries de dos números menors que 100. Les sumes han d'aparèixer a la pantalla en forma de taula i han d'estar distribuïdes en 4 fileres de 3 sumes cada una.

3. Presentar, centrades a la pantalla, 20 notes musicals (do, re, mi, fa, sol, la, si) escrites a l'atzar i una a continuació de l'altra.

4. Escriure una llista indefinida d'entrepans diferents formats aleatòriament a partir d'una llista de farcits i una altra de verdures.

Ex.: entrepà de pernil amb enciam
entrepà de formatge amb tomàquet
....

5. Dibuixar una línia diagonal amb un número variable d'asteriscos. Feu variables també la fila i la columna a partir de la qual es dibuixa. Ha de ser, per tant, un procediment amb tres paràmetres.

6. Escriure la taula de multiplicar del número que entra l'operador, procurant que els números quedin ben alineats.

7. Mirar si un número o una paraula és o no cap-i-cua. S'enténque un número o una paraula és cap-i-cua quan llegit del dret o del revés és el mateix.

8. Donat el dia i el mes del naixement d'una persona, esbrinar el seu signe de zodíac.

9. Calcular el nombre de bitllets i monedes de cada classe que calen per pagar a tots els treballadors d'una empresa, havent entrat els diferents sous dels treballadors.

10. Milloreu el programa proposat anteriorment (passar una data de la forma 20-6-89 a la forma 20 de juny de 1989), en el sentit que la data que s'entra pot tenir de 6 a 8 caràcters. Es a dir, si el número del dia o del més és inferior a 10 només es posa un caràcter. (Suggeriment: cal esbrinar en primer lloc on estan situats els guions).

B) Llistes: Dissenyeu l'algorisme i escriviu el corresponent programa per a la resolució dels següents problemes sobre el tractament de llistes:

11. Construir una funció que retorni el valor màxim dels elements d'una llista.

12. Esbrinar si els components d'una llista de números estan o no ordenats de forma creixent.

13. Escriviu un programa que en entrar dos cognoms els ordeni alfabèticament. Utilitzeu la funció menor :p1 :p2 proposada a la pràctica 11.

14. Dissenyeu un algorisme per validar una data entrada des del teclat de la forma dd/mm/aa o dd-mm-aa.

15. El mateix per a una hora de la forma hh:mm:ss.

16. Sumar dos números molt grossos (per exemple, 20 xifres) tractant-los com a paraules.

17. Recompte del nombre d'elements d'una llista que són menors que zero, iguals a zero i majors que zero.

18. Passar un número donat en segons a: hores, minuts i segons i escriure'l de la forma hh:mm:ss.

19. Passar un número donat de la forma hh:mm:ss, a segons.

20. Ordenar els valors d'una llista.

21. Afegir un element a una llista ordenada.

22. Escriviu un algorisme que admeti una frase com a entrada i indiqui les vegades que conté un determinat caràcter.

23. Procediment per posar un element dins d'una llista en una posició donada: afegeix n element llista

24. Procediment que retorna la posició de la primera aparició d'un element dins una llista: posició.llista element llista

25. Procediment que retorna els elements d'una llista a partir d'una certa posició: cua.llista n llista

26. Procediment que retorna els elements d'una llista compresos entre dues posicions donades: subllista m n llista

C) Mates: Dissenyeu l'algorisme i escriviu el corresponent programa per a la resolució dels següents problemes de caire matemàtic:

27. Calcular els dies que falten per acabar l'any a partir d'una data que s'entra en la forma dd/mm/aa.

28. Comptar els dies que hi ha entre dues dates entrades per l'operador.

29. Sumar temps expressat en la forma hh:mm:ss.

30. Un any es considera de traspàs si és múltiple de 4, excepte els múltiples de 100 que no ho són de 400. Donat un any, esbrinar (segons aquesta norma) si és o no de traspàs.

31. Escriure un algorisme que imprimeixi els n primers números de la sèrie de Fibonacci definida com:

$$a_0=0,\ a_1=1,\ a_n=a_{n-2}+a_{n-1} \qquad n > 1$$

32. Mirar si un número enter positiu donat és o no primer.

33. Determinar els valors màxim, mínim i la mitjana aritmètica d'una sèrie de números guardats en una llista.

34. Dissenyeu un programa per imprimir el Triangle de Pascal.

35. Procediment que escrigui tots els números cap-i-cua menors que 1000. Utilitzeu la funció proposada anteriorment.

D) Projectes: Dissenyeu l'algorisme i escriviu el corresponent programa per a la resolució dels següents problemes de caire general:

36. Programa que pregunti a l'usuari el nom d'un continent qualsevol i indiqui si és correcte o no.

37. Construir una funció que, donada una paraula, retorni el seu plural. Es pot millorar la solució trobada de manera que faci plurals més "complicats".

38. Procediment que, de forma aleatòria, pregunti la traducciód'un color donat en català. Un cop s'hagi preguntat un color s'ha d'eliminar de la llista perquè no el torni a preguntar.

39. Millorar l'exercici 1 en el sentit de dissenyar un programa capaç de conjugar:

a) qualsevol temps d'un verb de la primera.

b) qualsevol temps de qualsevol verb regular.

40. Programa traductor que disposi d'una llista de parelles de paraules (una, traducció de l'altra), pregunti a l'usuari la traducció d'una paraula escollida a l'atzar i indiqui si la resposta és o no correcta. També es pot fer que l'ordinador doni la resposta correcta en cas d'error.

41. Programa en què l'usuari tracta d'endevinar un número pensat de forma aleatòria per l'ordinador. Aquest indica si el número entrat per l'usuari és igual (fi del joc), inferior o superior al pensat per la màquina. Es pot posar un límit al nombre d'oportunitats que té la persona que juga

42. Algorisme per passar un número escrit en xifres decimals a xifres romanes.

43. Programa que generi frases aleatòries que tenen un mateix esquema.

44. Programa per passar diferents tipus de moneda (francs francesos, francs suïssos, marcs, lires, lliures, dòlars, etc.), segon la petició de l'usuari, a pessetes. Tota la informació necessària guardada en llistes.

45. Laberint: programa que dibuixi un laberint i després la tortuga, mitjançant sensors que detecten les parets, intenti sortir-ne.

46. Doctor: programa que simula un psicoanalista i que sigui capaç de portar una conversa amb l'usuari.

47. Diccionari (llista que creix): programa que retorni la traducció, en una determinada llengua, d'una paraula que entra l'usuari. Si no la coneix, que sigui capaç d'afegir-la a les seves llistes de tal manera que en una propera ocasió la respongui correctament.

48. Animal: programa que demana a l'usuari que es pensi un animal. Després l'ordinador fent preguntes intenta endevinar-lo. Si no ho aconsegueix demana el nom de l'animal i alguna característica que el permeti diferenciar d'altres animals que ja coneix. Es a dir, és un programa que va augmentant la seva base de coneixement.